

Autor: Rafael Alberto Moreno Parra

# Capacitándose en JAVASCRIPT

2020

# Contenido

Otros libros del autor .....	8
Página web del autor y canal en Youtube .....	9
Sitio en GitHub .....	9
Licencia de este libro .....	10
Licencia del software .....	10
Marcas registradas.....	10
Introducción .....	11
Iniciando.....	12
Navegadores .....	12
Editores de texto.....	12
Código fuente.....	12
Comentarios en el código .....	17
El tradicional “Hola Mundo” .....	18
Uso de variables.....	22
Operaciones matemáticas .....	24
Asignación .....	25
Operaciones con cadenas .....	26
Impresión de caracteres propios del idioma español.....	29
Constantes matemáticas .....	32
Funciones trigonométricas .....	33
Funciones Matemáticas .....	34
Captura de datos tipo texto por pantalla .....	36
Captura de datos tipo numérico por pantalla .....	37
Ejemplos de cálculos.....	39
Corrigiendo errores en JavaScript.....	40
Herramienta de Google para programar en JavaScript.....	42
Validación en línea del código .....	43
Sí condicional .....	46
Operadores lógicos: Conjunción (Y, &&) y Disyunción (O,   ) .....	48
Uso del switch .....	49
Ciclos. Uso del “for” .....	50

Ciclo while .....	55
Ciclo do - while.....	58
Uso del break .....	61
Uso del continue .....	62
Ciclos anidados y salir de estos.....	63
Funciones o subrutinas .....	65
Ejemplos de uso de la librería .....	72
Método de Bisección .....	82
Ámbito (scope) de las variables y funciones.....	83
Funciones recursivas.....	87
Uso de eval como evaluador de expresiones algebraicas .....	90
Números aleatorios .....	92
Distribución Normal .....	93
Distribución Triangular.....	94
Distribución Uniforme .....	95
Manejo de errores con números .....	96
isNaN .....	96
isFinite .....	98
Try...catch .....	100
Arreglos unidimensionales o vectores.....	102
Borrar elementos con splice( ).....	105
Operaciones con arreglos .....	106
Ordenación de arreglo unidimensional .....	108
Funciones genéricas para arreglos .....	111
Retornar arreglos desde funciones.....	113
Implementación de algoritmos de ordenación .....	114
Algoritmo de burbuja.....	114
Algoritmo de ordenación por selección.....	117
Algoritmo de ordenación por inserción .....	118
Algoritmo de ordenación QuickSort .....	119
Algoritmo de ordenación MergeSort.....	120
Manejo de Fechas .....	122

Manejo de cadenas o strings .....	124
Ejemplos de programas usando cadenas .....	127
Invertir Cadena.....	127
Quitar los espacios de una cadena .....	128
Quitar las vocales de una cadena .....	129
Quitar caracteres no permitidos.....	130
Un sencillo cifrado / descifrado .....	131
Un sencillo cifrado / descifrado con clave de cifrado .....	133
Sumar largos números enteros almacenados en cadenas .....	135
Arreglos bidimensionales.....	136
Ejemplos de programas usando arreglos bidimensionales .....	140
Poner una torre al azar en un tablero de ajedrez y mostrar su ataque .....	140
Poner un Rey al azar en un tablero de ajedrez y mostrar su ataque .....	142
Poner un alfil al azar en un tablero de ajedrez y mostrar su ataque.....	144
Poner una Reina al azar en un tablero de ajedrez y mostrar su ataque .....	146
Ruta del menor costo.....	148
Resolver Sudokus .....	151
Poner los barcos en el juego batalla naval .....	154
Funciones que reciben distinto número de parámetros .....	157
Funciones que reciben a su vez funciones por parámetros .....	159
Operaciones de bit.....	161
Convertir un entero a su representación binaria .....	161
Operaciones OR, AND y XOR directas.....	162
Multiplicar usando operaciones de bit .....	164
Dividir usando operaciones de bit .....	165
Intercambiar valores usando operaciones de bit .....	166
Programación orientada a objetos .....	167
Definiendo clases, constructores y atributos .....	167
Definiendo métodos .....	168
Métodos y atributos .....	169
Atributos y métodos privados .....	170
Getters y Setters .....	171

Herencia .....	173
Herencia y constructor.....	174
Polimorfismo.....	176
Uso de typeof para detectar el tipo de dato de una variable .....	178
Árbol binario .....	180
Recorrido en pre-orden .....	181
Recorrido en in-orden.....	183
Recorrido en post-orden.....	184
Ordenación usando un árbol binario .....	185
Lista de objetos. Pilas.....	187
Lista de objetos.....	189
Instrucciones que ejecutan controlando el tiempo.....	190
Uso de JSON .....	193
Control sobre la página HTML .....	199
Captura el evento de dar clic en un <div> .....	201
Captura el evento de dar doble clic en un <div> .....	203
Captura el evento de pasar el cursor del ratón por encima de un <div> .....	204
Captura el evento de mover el cursor del ratón fuera del <div> .....	205
Captura el evento apenas cargue la página.....	206
Captura el evento cuando de clic al interior del <div> .....	207
Captura el evento al soltar el botón del ratón dentro del <div> .....	208
Captura el evento de mover el cursor del ratón.....	209
Captura el evento cuando se cambia el tamaño de la ventana .....	210
Captura el evento de dar “submit” en un formulario .....	211
Captura el evento de entrar a la caja de texto .....	212
Captura el evento al salir de una caja de texto .....	213
Captura el evento al presionar una tecla dentro de una caja de texto .....	214
Captura el evento cuando suelta la tecla .....	216
Captura el evento de cambio del texto en una caja de texto.....	217
Mostrar el código de la tecla presionada .....	218
Pregunta antes de ir a la página enlazada .....	219
DOM (Document Object Model).....	220

Cambiar atributos de un objeto.....	220
Cambiar el color de fondo de un <div>.....	221
Hacer visible o no un <div> .....	222
Poner o quitar texto de un <div> .....	224
Cambiar el aspecto del objeto cuando está seleccionado .....	226
Validar valores al saltar a otro objeto.....	228
Contar objetos .....	229
Obtener lista de objetos .....	230
Obtener determinado objeto de una lista.....	231
Obtener lista de determinado tipo de objetos y ver el valor de sus atributos .....	232
Obtener la lista de determinado tipo de objetos y cambiar sus atributos.....	233
Mostar la ubicación del documento .....	235
Cambiar el estilo CSS de un determinado objeto .....	236
Mostrar la codificación del documento .....	238
Mostrar el título del documento .....	239
Saltos de línea sin usar   .....	240
Mostrar la fecha de modificación del documento .....	241
Mostrar el tipo de documento.....	242
Mostrar el dominio del documento.....	243
Mostrar el estado del documento .....	244
Mostrar las dimensiones del documento .....	245
Mostrar el número de enlaces.....	247
Mostrar hacia donde apuntan los enlaces.....	248
Crear botones en tiempo de ejecución .....	251
Crear <p> en tiempo de ejecución.....	253
Crear nuevos <p> y contarlos en tiempo de ejecución .....	255
Contar el número de objetos en el documento .....	257
Contar el número de formularios en el documento .....	258
Contar el número de imágenes del documento .....	259
Describir los objetos de la página .....	261
Abrir una ventana .....	264
Abrir un conjunto de ventanas .....	265

Un formulario que ejecuta un algoritmo local al enviar la información .....	267
Valida un correo con una expresión regular.....	270
Valida una URL con expresiones regulares .....	272
Gráficos .....	274
Dibujar líneas .....	276
Dibujar círculos .....	292
Dibujar curvas .....	294
Curva de Bézier .....	296
Rectángulo y gradiente de color .....	299
Trabajando con imágenes.....	306
Convertir imagen a escala de grises .....	309
Dibujar letras.....	311
Sombras .....	315
Semitransparente .....	316
Referencias.....	317

## Otros libros del autor

Libro 13: "Algoritmos Genéticos". En Colombia 2020. Págs. 62. Libro y código fuente descargable en: <https://github.com/ramsoftware/LibroAlgoritmoGenetico2020>

Libro 12: "Redes Neuronales. Segunda Edición". En Colombia 2020. Págs. 108. Libro y código fuente descargable en: <https://github.com/ramsoftware/LibroRedNeuronal2020>

Libro 11: "Capacitándose en JavaScript". En Colombia 2020. Págs. 317. Libro y código fuente descargable en: <https://github.com/ramsoftware/javascript>

Libro 10: "Desarrollo de aplicaciones para Android usando MIT App Inventor 2". En Colombia 2016. Págs. 102. Ubicado en: <https://openlibra.com/es/book/desarrollo-de-aplicaciones-para-android-usando-mit-app-inventor-2>

Libro 9: "Redes Neuronales. Parte 1.". En Colombia 2016. Págs. 90. Libro descargable en: <https://openlibra.com/es/book/redes-neuronales-parte-1>

Libro 8: "Segunda parte de uso de algoritmos genéticos para la búsqueda de patrones". En Colombia 2015. Págs. 303. En publicación por la Universidad Libre – Cali.

Libro 7: "Desarrollo de un evaluador de expresiones algebraicas. **Versión 2.0**. C++, C#, Visual Basic .NET, Java, PHP, JavaScript y Object Pascal (Delphi)". En: Colombia 2013. Págs. 308. Ubicado en: <https://openlibra.com/es/book/evaluador-de-expresiones-algebraicas-ii>

Libro 6: "Un uso de algoritmos genéticos para la búsqueda de patrones". En Colombia 2013. En publicación por la Universidad Libre – Cali.

Libro 5: Desarrollo fácil y paso a paso de aplicaciones para Android usando MIT App Inventor. En Colombia 2013. Págs. 104. Estado: Obsoleto (No hay enlace).

Libro 4: "Desarrollo de un evaluador de expresiones algebraicas. C++, C#, Visual Basic .NET, Java, PHP, JavaScript y Object Pascal (Delphi)". En Colombia 2012. Págs. 308. Ubicado en: <https://openlibra.com/es/book/evaluador-de-expresiones-algebraicas>

Libro 3: "Simulación: Conceptos y Programación" En Colombia 2012. Págs. 81. Ubicado en: <https://openlibra.com/es/book/simulacion-conceptos-y-programacion>

Libro 2: "Desarrollo de videojuegos en 2D con Java y Microsoft XNA". En Colombia 2011. Págs. 260. Ubicado en: <https://openlibra.com/es/book/desarrollo-de-juegos-en-2d-usando-java-y-microsoft-xna>. ISBN: 978-958-8630-45-8

Libro 1: "Desarrollo de gráficos para PC, Web y dispositivos móviles" En Colombia 2009. ed.: Artes Gráficas Del Valle Editores Impresores Ltda. ISBN: 978-958-8308-95-1 v. 1 págs. 317

Artículo: "Programación Genética: La regresión simbólica". Entramado ISSN: 1900-3803 ed.: Universidad Libre Seccional Cali v.3 fasc.1 p.76 - 85, 2007

## Página web del autor y canal en Youtube

Investigación sobre Vida Artificial: <http://darwin.50webs.com>

Canal en Youtube: <http://www.youtube.com/user/RafaelMorenoP> (dedicado al desarrollo en C#)

## Sitio en GitHub

El código fuente se puede descargar en <https://github.com/ramsoftware/>

Licencia de este libro



Licencia del software

Todo el software desarrollado aquí tiene licencia LGPL "Lesser General Public License"



Marcas registradas

En este libro se hace uso de las siguientes tecnologías registradas:

Microsoft ® Windows ® Enlace: <http://windows.microsoft.com/en-US/windows/home>

## Introducción

En el desarrollo de aplicaciones Web, en el lado del cliente, JavaScript [1] [2] es el lenguaje de programación con una aplastante mayoría. Sólo es ver el siguiente gráfico a julio de 2020 de cómo está repartido los lenguajes de programación en el cliente [3]:

El enlace es: [https://w3techs.com/technologies/overview/client\\_side\\_language/all](https://w3techs.com/technologies/overview/client_side_language/all)

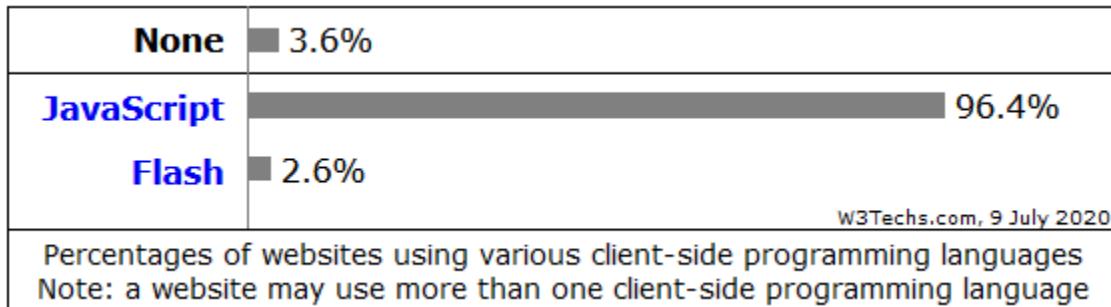


Ilustración 1: Lenguajes de programación para aplicaciones Web en el lado del cliente

JavaScript es el lenguaje de facto, además, le sumamos que Flash [4] es una tecnología que será abandonada por la empresa que la creó [5].

Es necesario programar en el lado del cliente para las siguientes tareas:

1. Validaciones de datos en el cliente, por ejemplo, que el usuario final sólo ingrese números en un determinado campo y evitar que ingrese letras.
2. Mostrar gráficos.
3. Mejorar la interactividad de la página web.
4. Ejecutar procesos en el lado del cliente.

El presente libro es sobre JavaScript basado en el estándar ECMAScript 6 - ECMAScript 2015 [6] [7]

## Iniciando

Empezar a programar en JavaScript es fácil y gratuito. Sólo requerimos tener un navegador moderno instalado en el PC (Microsoft Edge, Mozilla Firefox, Opera, Google Chrome, Vivaldi) y un editor de texto (recomendado Notepad++ que es gratuito). Este manual mostrará capturas de pantalla en el ambiente de Microsoft Windows 10, pero lo expuesto es aplicable en otros sistemas operativos porque JavaScript es multiplataforma.

## Navegadores

Enlace para Mozilla Firefox: <https://www.mozilla.org/es-ES/firefox/new/>

Enlace para Opera: <https://www.opera.com/es-419>

Enlace para Google Chrome: <https://www.google.com.mx/chrome/>

Enlace para Vivaldi: <https://vivaldi.com/es/>

Debe considerar el soporte que da a JavaScript los navegadores, ver más en:

[https://www.w3schools.com/js/js\\_versions.asp](https://www.w3schools.com/js/js_versions.asp) [8]

## Editores de texto

Enlace para Notepad++: <https://notepad-plus-plus.org/download/v7.6.2.html>

## Código fuente

El código fuente se puede descargar de: --- PONER AQUÍ SITIO DE GITHUB O PAGINA WEB ---

Donde se muestre código fuente en JavaScript, se mostrará la palabra “Directorio N”. Significa que cuando descomprima el archivo descargable, hallará ese código fuente (un archivo HTML, en algunos casos scripts .JS) en ese directorio o carpeta. Es mejor referirse directamente a ese código fuente que copiar y pegar el código de este documento donde se corre el riesgo de perder el formato.

Para escribir código fuente abrimos Notepad++, obtenemos un documento en blanco, los menús están en inglés, así que, si nos es más cómodo trabajar con el software en español, lo configuramos así:

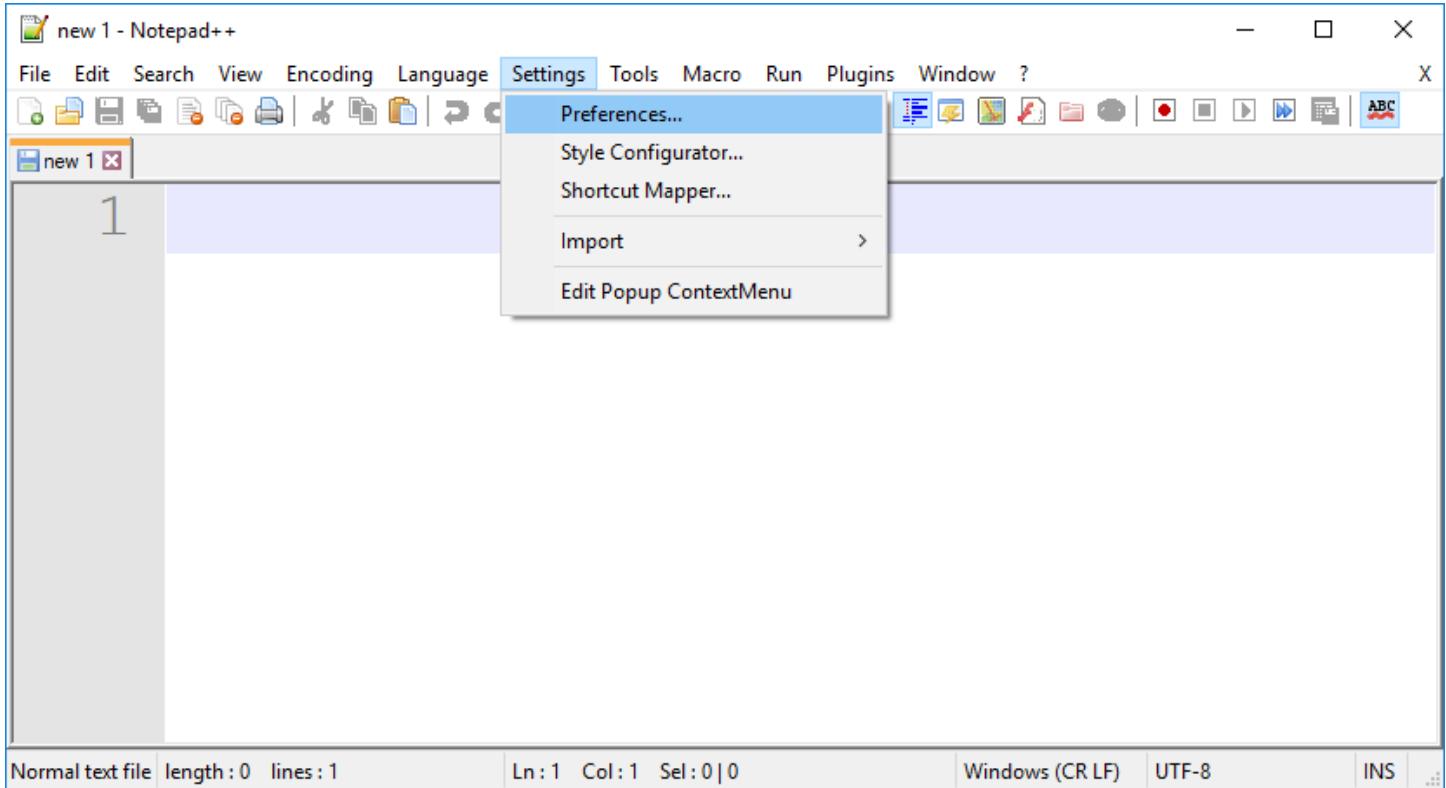


Ilustración 2: Cambiar la configuración del Notepad++

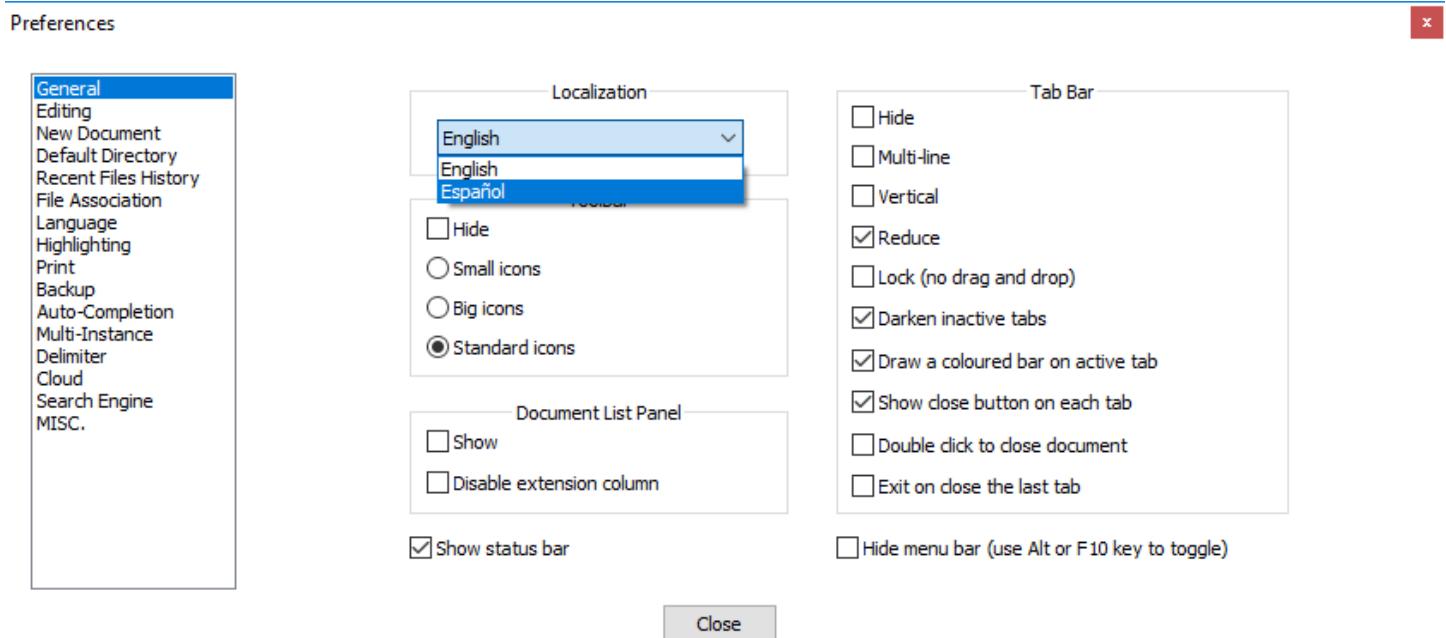


Ilustración 3: Cambiando a idioma en español

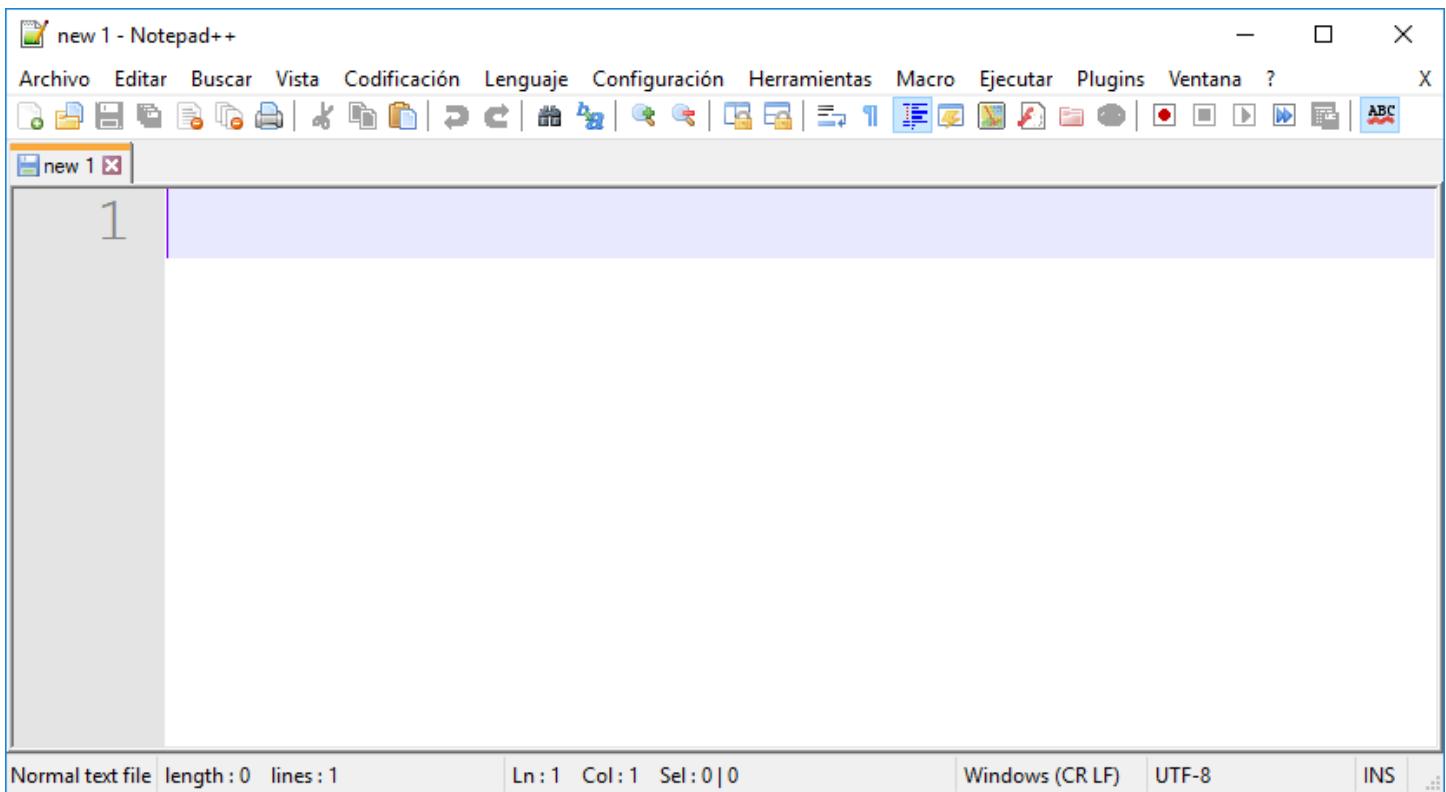


Ilustración 4: Notepad++ en español

Configuramos el Notepad++ para trabajar con JavaScript, para eso vamos por la opción Lenguaje → H → HTML

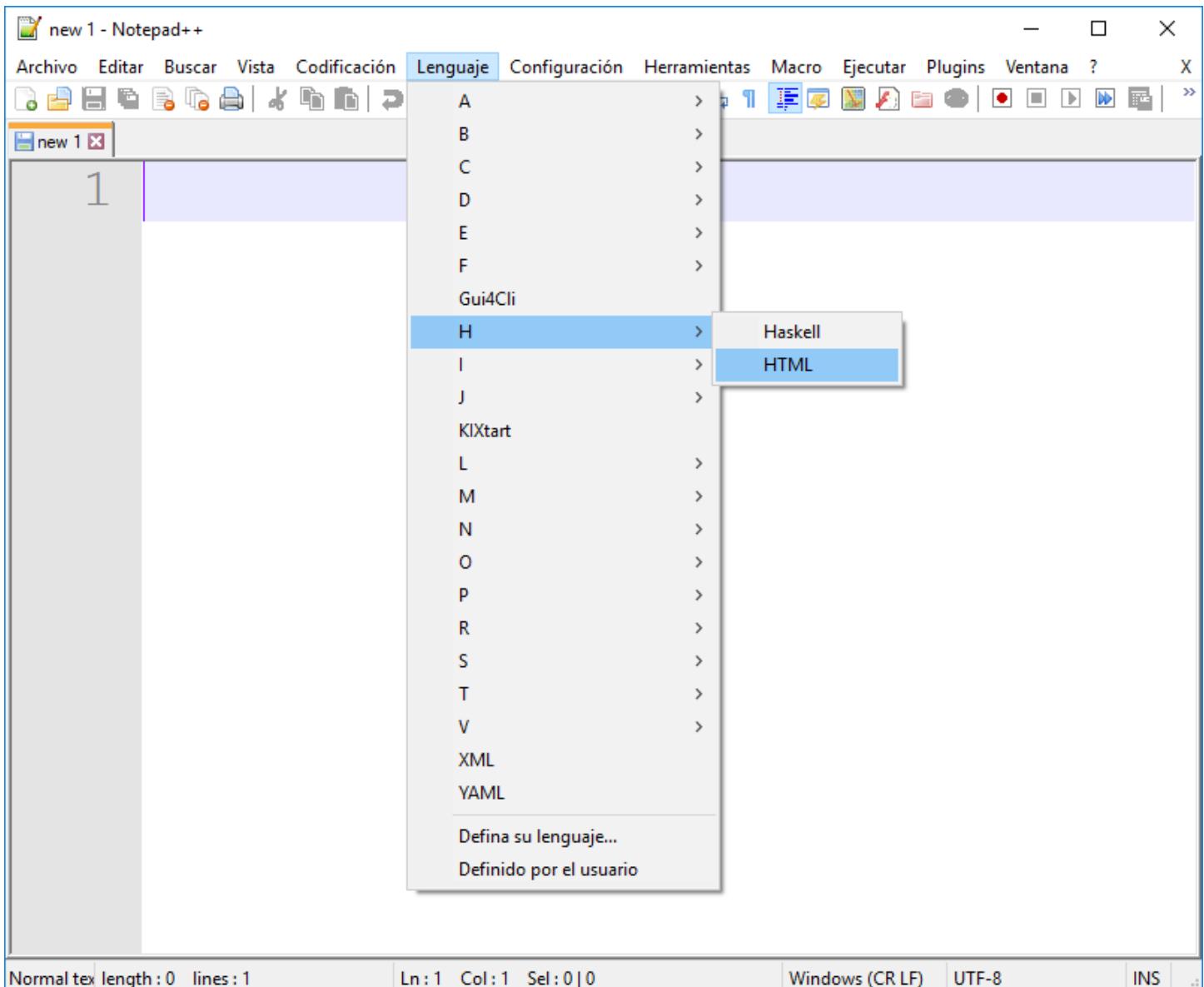


Ilustración 5: Configuramos Notepad++ para editar archivos HTML y JavaScript

¡Un momento! ¿Por qué por HTML y no JavaScript que está un poco más abajo en el menú? Nuestros programas en JavaScript, para empezar, deben estar dentro de una página HTML que es leída por el navegador y el código en JavaScript es ejecutado por el mismo navegador. Más adelante, separaremos el código HTML del de JavaScript (en lecciones más avanzadas).

Escribimos el esqueleto, lo mínimo necesario para hacer un programa en JavaScript:

The screenshot shows a Notepad++ window with the title bar "new 1 - Notepad++". The menu bar includes Archivo, Editar, Buscar, Vista, Codificación, Lenguaje, Configuración, Herramientas, Macro, Ejecutar, Plugins, Ventana, and Help. The toolbar has various icons for file operations like Open, Save, Print, and Find. The main editor area contains the following code:

```
<!DOCTYPE HTML><html><head><script>
</script></head></html>
```

The first line, <!DOCTYPE HTML>, is highlighted in blue. The entire code block is selected, indicated by a red selection bar on the left and a light purple background for the text area.

At the bottom, the status bar shows "Hyper Text length : 64 lines : 4", "Ln : 4 Col : 24 Sel : 0 | 0", "Windows (CR LF)", "UTF-8", and "INS".

Ilustración 6: Plantilla básica HTML5 para escribir programas en JavaScript

Este es el código:

Directorio 001

```
<!DOCTYPE HTML><html><head><script>
</script></head></html>
```

La etiqueta <!DOCTYPE HTML> informa al navegador que se usará la versión más reciente de HTML, es decir, HTML 5.

Cada etiqueta en HTML se abre y se cierra, ejemplo, <html> y su cierre </html>

El código en JavaScript se ubica entre <script> y su cierre </script>

## Comentarios en el código

Los comentarios son clave en la documentación del código, hacen la diferencia entre un software en que podamos hacer mantenimiento y software que hay que desechar. Debemos usarlos bastante y bien. Similar a C++, si el comentario es de una sola línea, se usa // pero si se requieren varias líneas se puede encerrar entre /\* y \*/ . Ver el código:

*Directorio 002*

```
<!DOCTYPE HTML><html><head><script>
//Comentario de una sola línea

/* Comentarios en
   varias
   líneas */

</script></head></html>
```

## El tradicional “Hola Mundo”

Con la instrucción document.write podemos mostrar mensajes en el navegador. Las instrucciones deben terminar en punto y coma (;

The screenshot shows a Notepad++ window with the title bar "new 1 - Notepad++". The menu bar includes Archivo, Editar, Buscar, Vista, Codificación, Lenguaje, Configuración, Herramientas, Macro, Ejecutar, Plugins, Ventana, and Help. The toolbar below has various icons for file operations like Open, Save, Print, and Find. The code editor contains the following HTML script:

```
1 <!DOCTYPE HTML><html><head><script>
2 //Imprimir una linea
3
4 document.write("Hola Mundo");
5
6</script></head></html>
```

The status bar at the bottom shows "Hyper Text length : 118 lines : 6 Ln : 4 Col : 30 Sel : 0 | 0 Windows (CR LF) UTF-8 INS".

Ilustración 7: Imprimir un texto

Guardamos el documento como un archivo HTML, presionamos el botón de guardar

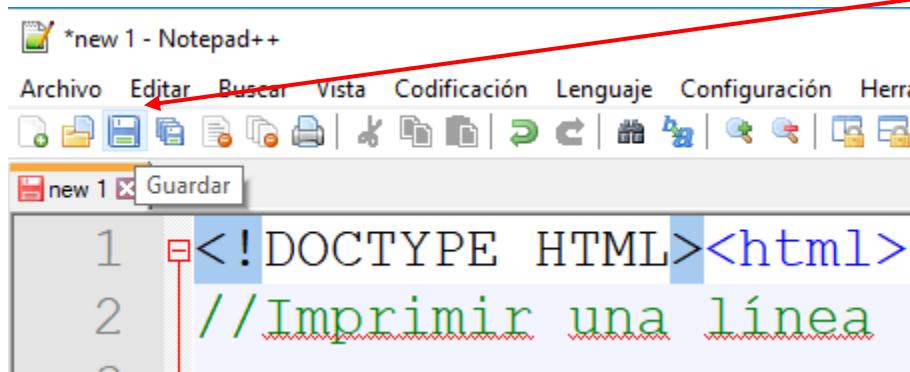


Ilustración 8: Botón de guardar

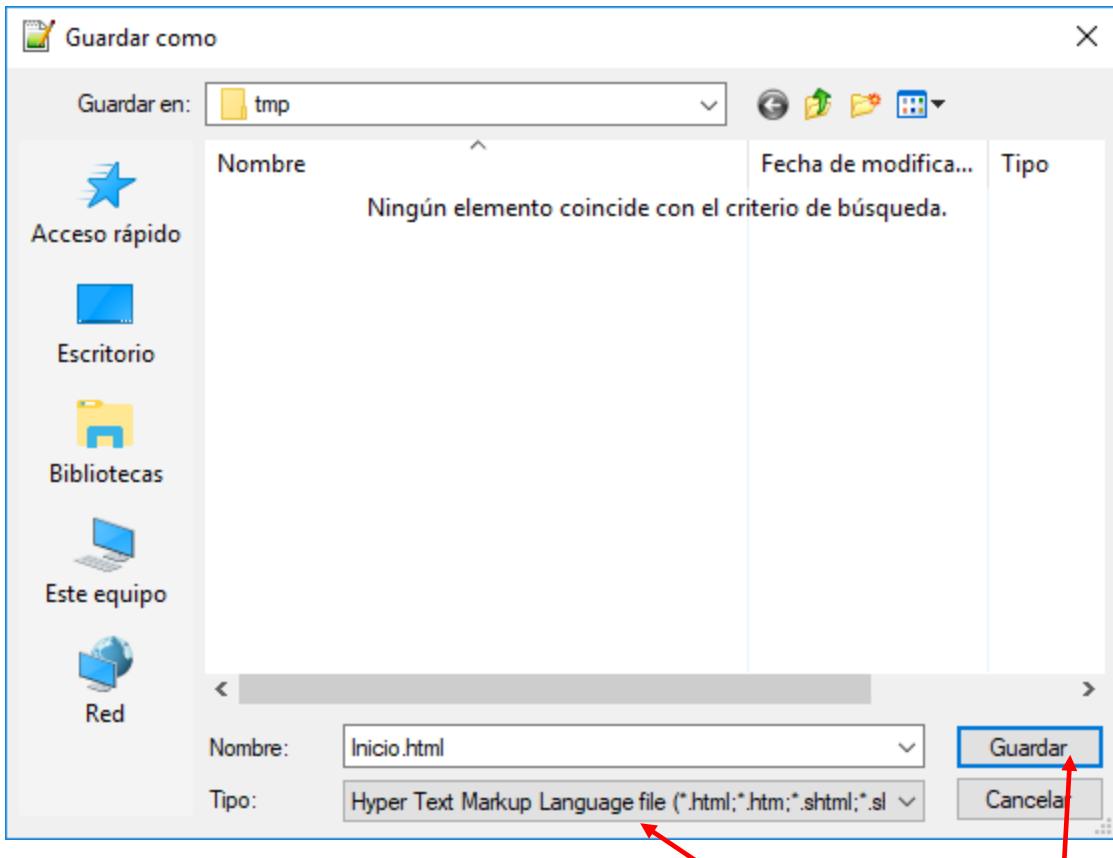


Ilustración 9: Guardar la página HTML en el PC

Debemos estar atentos que en Tipo: diga Hyper Text Markup Language file y lo guardamos.

```
<!DOCTYPE HTML><html><head><script>
//Imprimir una línea
</script></head></html>
```

Ilustración 10: Debemos fijarnos que el archivo tenga extensión .html

Hay que fijarnos que la extensión del archivo sea .html

El siguiente paso es ejecutar nuestra página Web, para ello hay que ir por Ejecutar

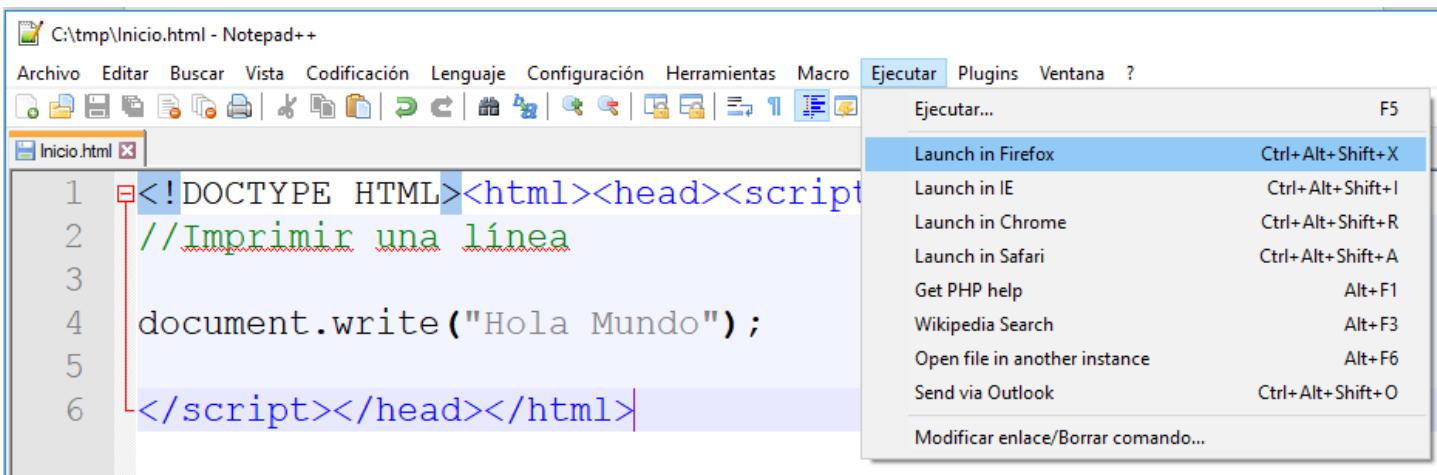
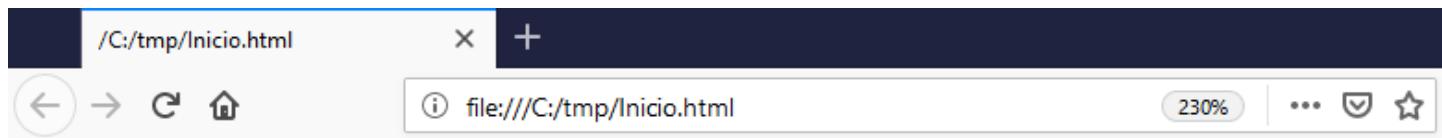


Ilustración 11: Inicia el navegador seleccionado para mostrar la página

Dependiendo de los navegadores que tengamos instalados escogemos la opción correcta. Por ejemplo, si tenemos Mozilla Firefox seleccionamos “Launch in Firefox” o si tenemos Google Chrome seleccionamos “Launch in Chrome”. Se lanza inmediatamente el navegador.



# Hola Mundo

Ilustración 12: La ejecución del programa en JavaScript en el navegador Mozilla Firefox

**¡Advertencia!** Debe tener guardada la página antes de ejecutarla en un navegador.

Este es el código:

Directorio 003

```
<!DOCTYPE HTML><html><head><script>
//Imprimir una línea

document.write("Hola Mundo");

</script></head></html>
```

Las comillas dobles pueden ser reemplazadas por comillas simples.

Directorio 003

```
<!DOCTYPE HTML><html><head><script>
//Imprimir una línea

document.write('Hola Mundo');

</script></head></html>
```

Y además podemos poner código HTML en su interior para hacer cambios en el texto

Directorio 003

```
<!DOCTYPE HTML><html><head><script>
//Imprimir una línea

document.write('<b>Hola Mundo</b>');

</script></head></html>
```

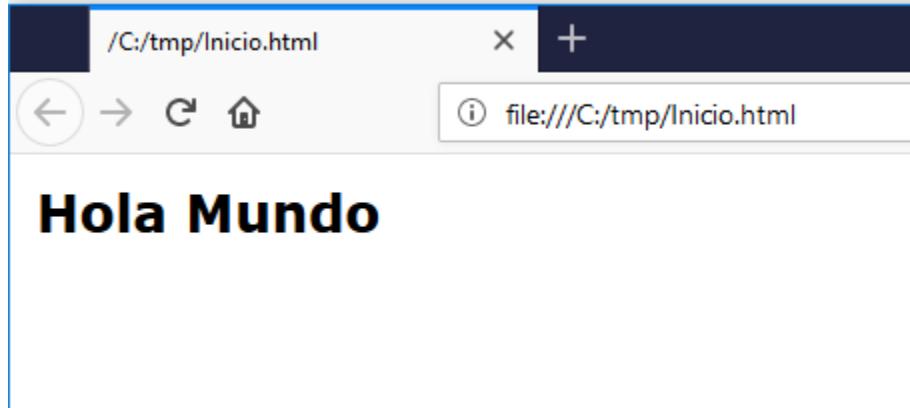


Ilustración 13: Mensaje en negrillas

## Uso de variables

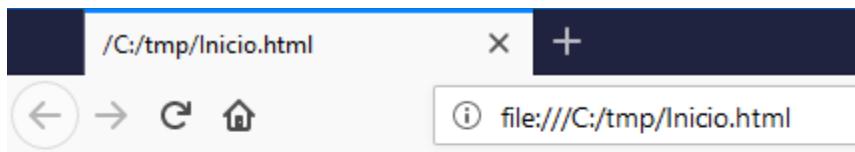
JavaScript declara las variables con la palabra reservada var. Es un lenguaje débilmente tipado, es decir, no definimos variables enteras, reales o de cadena. JavaScript detecta el tipo de dato cuando se le asigna el primer valor a la variable. Si se le asigna un texto, se considera que es una variable tipo texto (string).

Directorio 004

```
<!DOCTYPE HTML><html><head><script>
//Variables en JavaScript
var valA = 15;
var cadena = "Esta es una prueba";
var aproxima = -7.126;

document.write(valA);
document.write("<br>");
document.write(cadena);
document.write("<br>");
document.write(aproxima);
</script></head></html>
```

La instrucción document.write("<br>"); es para que haya un salto entre cada dato impreso.



15  
Esta es una prueba  
-7.126

Ilustración 14: Impresión del valor de cada variable

Podemos variar el código para impresión y que quede en una sola línea

Directorio 004

```
<!DOCTYPE HTML><html><head><script>
//Variables en JavaScript
var valA = 78;
var cadena = "Cadena de texto";
var aproxima = -90.23;

document.write(valA + "<br>");
document.write(cadena + "<br>");
document.write(aproxima);
</script></head></html>
```

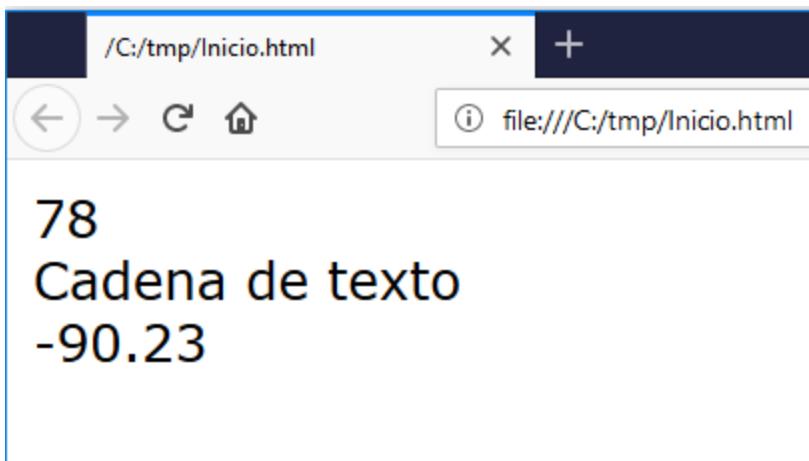


Ilustración 15: Impresión de valor de cada variable

O todo en una línea

Directorio 004

```
<!DOCTYPE HTML><html><head><script>
//Variables en JavaScript
var val = -901;
var cad = "Manual de JavaScript";
var aprox = 864.5;

document.write(val + "<br>" + cad + "<br>" + aprox);
</script></head></html>
```

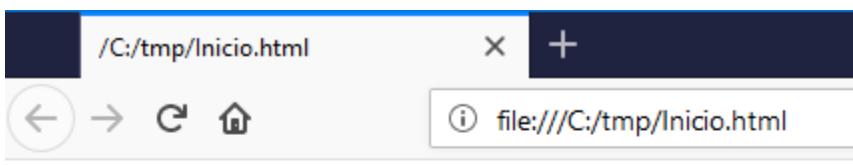


Ilustración 16: Impresión de valor de cada variable

## Operaciones matemáticas

Están disponibles suma(+), resta(-), multiplicación(\*), división(/), potencia(\*\*) y división modular(%).

Recordemos que la división modular retorna el residuo de la división. Este es el código:

Directorio 005

```
<!DOCTYPE HTML><html><head><script>
//Aritmética
var valA = 12 + 45;
var valB = 55 - 89;
var valC = 7 * 8;
var valD = 21 / 3;
var valE = 5 ** 3; //5 al cubo
var valF = 5671 % 2; //Determina el residuo

document.write(valA + "<br>" + valB + "<br>");
document.write(valC + "<br>" + valD + "<br>");
document.write(valE + "<br>" + valF + "<br>");
</script></head></html>
```

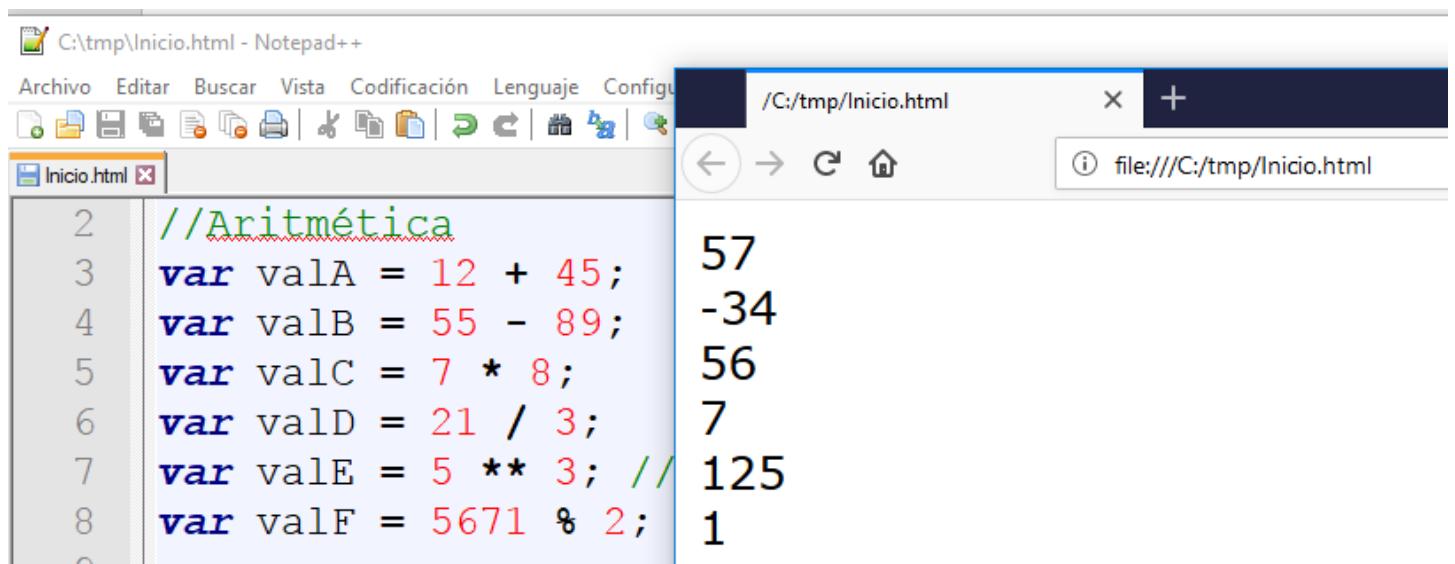


Ilustración 17: Resultado de la ejecución de las operaciones matemáticas

## Asignación

Se utiliza el operador = para asignar un valor a una variable. Tiene algunas variaciones para acotar operaciones

Directorio 006

```
<!DOCTYPE HTML><html><head><script>
//Asignación
var valA = 8;
var valB = valA * 2;
document.write(valA + " y " + valB + "<br>");

valA += 10; //Sumarle 10
valB -= 5; //Restarle 5
document.write(valA + " y " + valB + "<br>");

valA *= 3; //Multiplique por 3
valB /= 2; //Divida entre 2
document.write(valA + " y " + valB + "<br>");

</script></head></html>
```

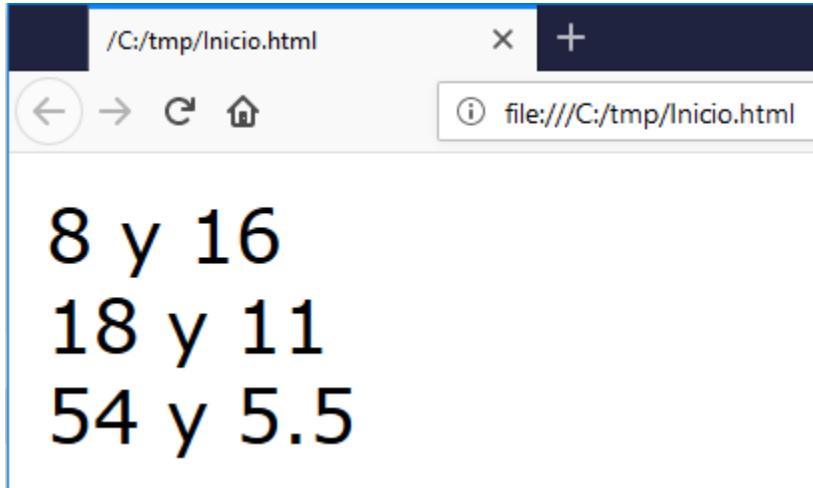


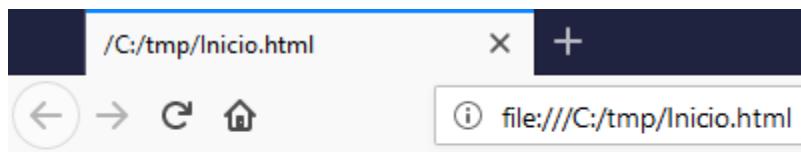
Ilustración 18: Combinando texto con valores

## Operaciones con cadenas

Se utiliza el operador + para concatenar cadenas.

Directorio 007

```
<!DOCTYPE HTML><html><head><script>
//Uso de cadenas
var txtA = "Una";
var txtB = "Cadena";
var txtC = txtA + " --- " + txtB;
document.write(txtC);
</script></head></html>
```



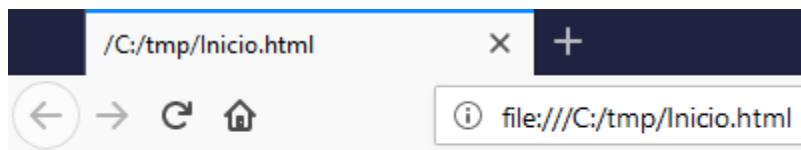
### Una --- Cadena

Ilustración 19: Resultado de la concatenación

¿Y qué sucede al combinar números y texto?

Directorio 007

```
<!DOCTYPE HTML><html><head><script>
//Uso de cadenas
var txtA = "Una";
var txtB = 45.98;
var txtC = txtA + " --- " + txtB;
document.write(txtC);
</script></head></html>
```



### Una --- 45.98

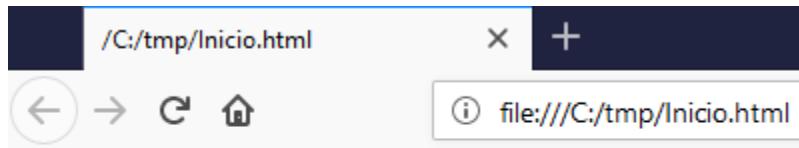
Ilustración 20: Se concatena texto con números

Otro cambio, ambas variables con números reales.

Directorio 007

```
<!DOCTYPE HTML><html><head><script>
//Uso de cadenas
var txtA = 27.6;
var txtB = 45.98;
var txtC = txtA + " --- " + txtB;
document.write(txtC);
</script></head></html>
```

Esto sucede:



27.6 --- 45.98

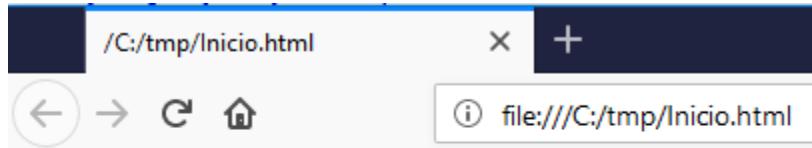
Ilustración 21: Se concatena números y texto

Pero si se hace este cambio (poner a sumar las variables con valor de número real y luego concatenar una cadena):

Directorio 007

```
<!DOCTYPE HTML><html><head><script>
//Uso de cadenas
var txtA = 7;
var txtB = 5;
var txtC = txtA + txtB + "aaaaa";
document.write(txtC);
</script></head></html>
```

Pasa esto:



12aaaaa

Ilustración 22: Primero se hace la operación matemática y luego la concatenación

## ¿Y si hacemos más operaciones matemáticas?

Directorio 007

```
<!DOCTYPE HTML><html><head><script>
//Uso de cadenas
var txtA = 7;
var txtB = 5;
var txtC = txtA + txtB + "aaaaa" + txtA + txtB;
document.write(txtC);
</script></head></html>
```

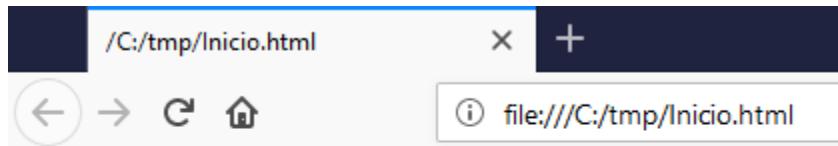
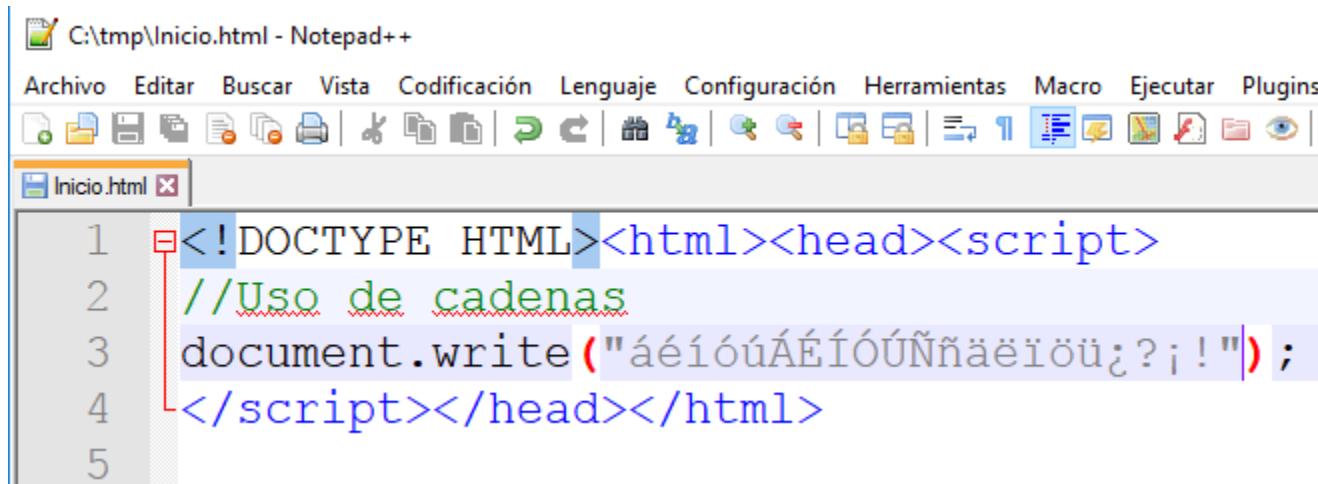


Ilustración 23: Se hace la operación matemática y el resto es concatenación

## Impresión de caracteres propios del idioma español

Queremos imprimir caracteres como tildes, Ñs, abre interrogante o abre admiración. El código a continuación:



The screenshot shows the Notepad++ interface with the file 'Inicio.html' open. The code is as follows:

```
1 <!DOCTYPE HTML><html><head><script>
2 //Uso de cadenas
3 document.write("áéíóúÁÉÍÓÚÑñäëöü¿¡!\"");
4 </script></head></html>
5
```

A red box highlights the opening tag of the script block.

Ilustración 24: Imprimir caracteres propios del idioma español

Y al ejecutarlo esto sucede en Google Chrome:

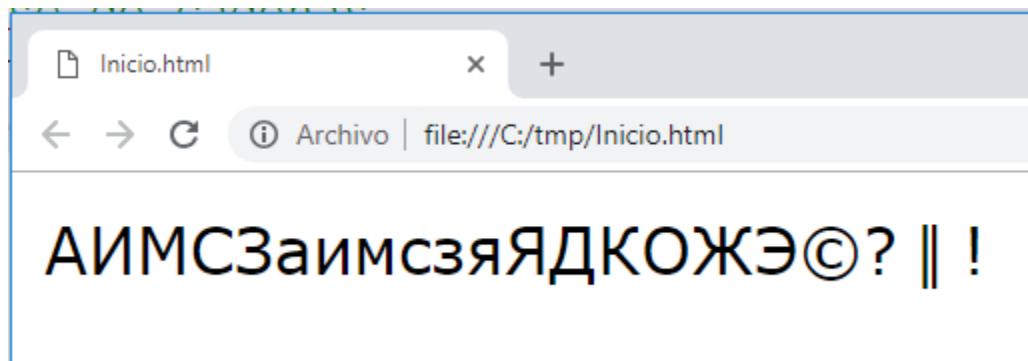


Ilustración 25: En Chrome puede pasar que no se impriman

Funciona correcto en Mozilla Firefox:

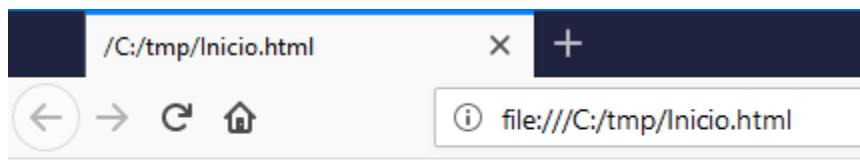


Ilustración 26: Funciona en Mozilla Firefox

¿Por qué en Google Chrome no se muestran correctamente los caracteres? Hay que fijarse en la codificación que tiene Notepad++ sobre el documento:

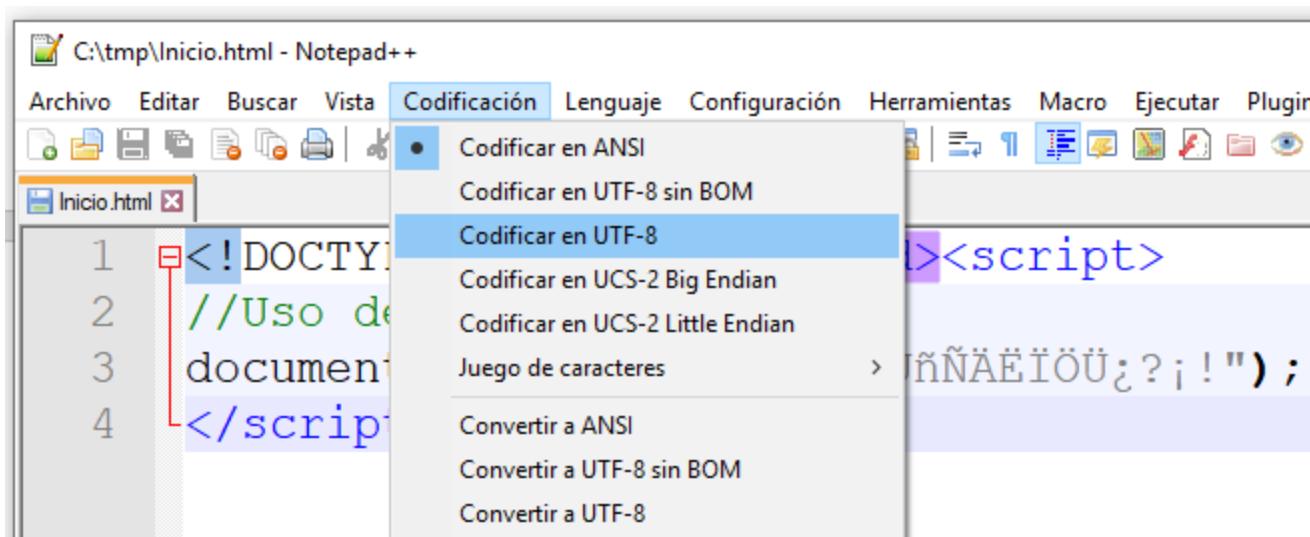


Ilustración 27: Ante un fallo así, verificamos la codificación. Usualmente está en "Codificar en ANSI"

El problema está en que la opción es “Codificar en ANSI”, luego el siguiente paso es dar clic a “Convertir a UTF-8”, guardar y probar de nuevo:

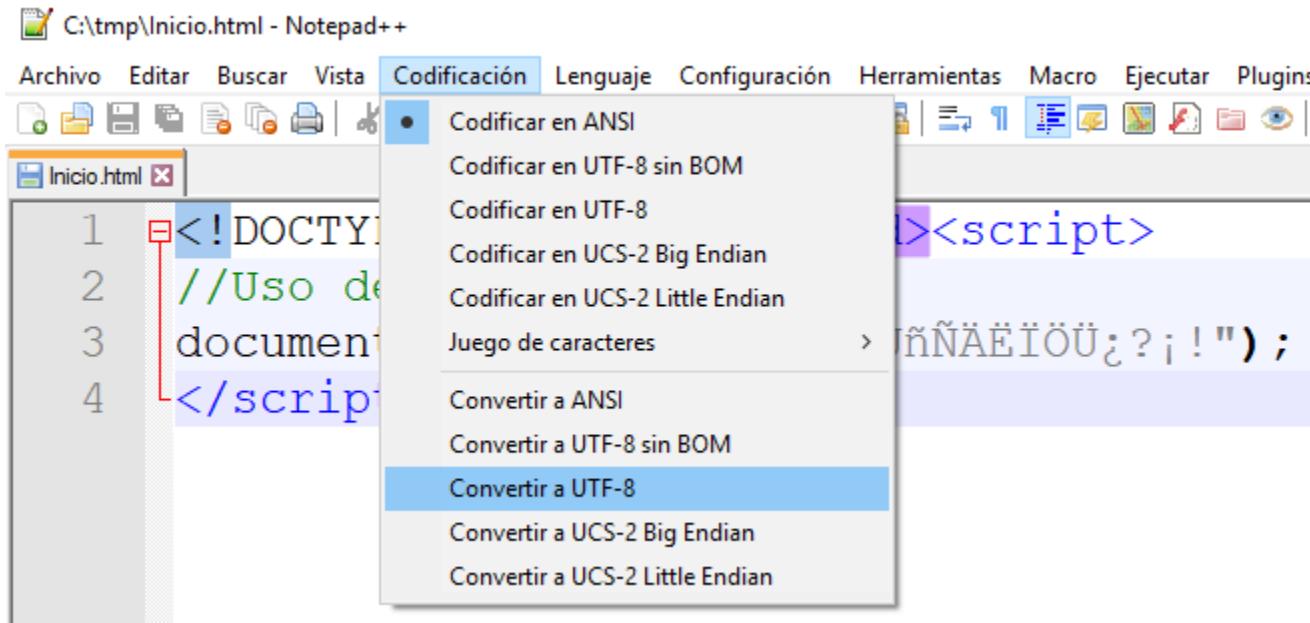


Ilustración 28: Seleccionamos la opción de "Convertir a UTF-8"

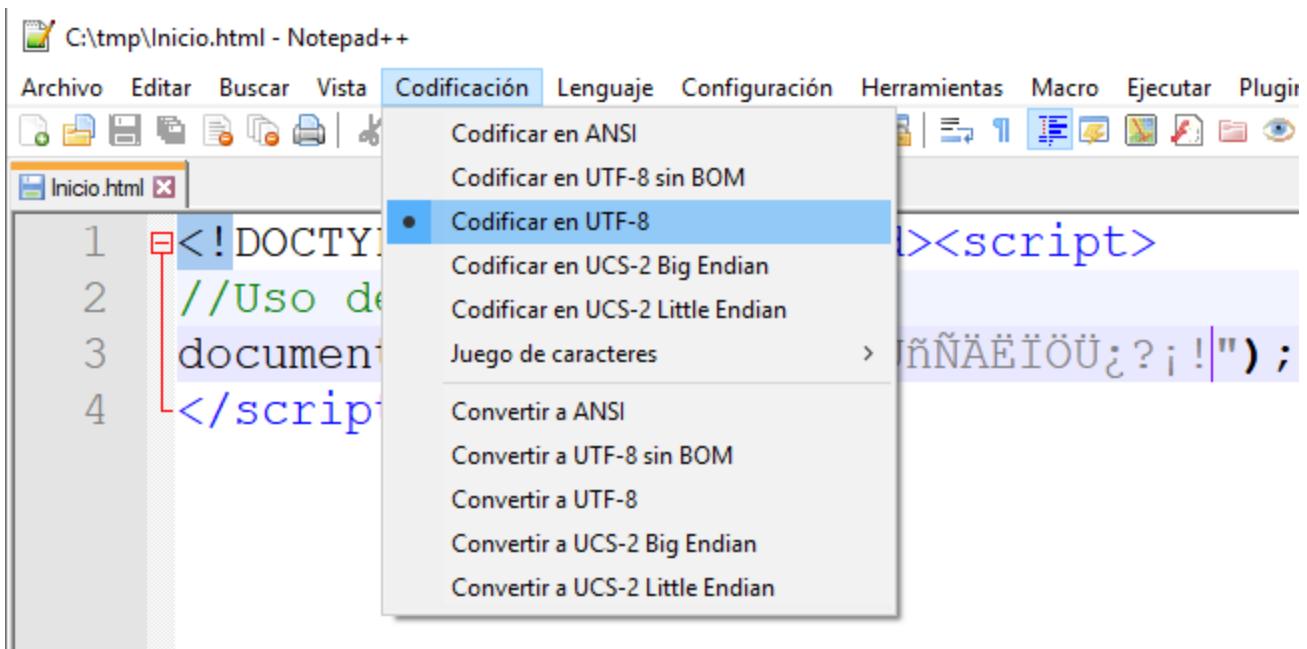
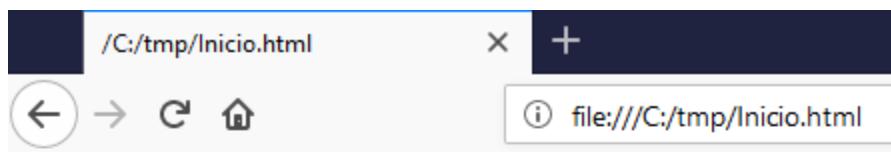


Ilustración 29: Despues de guardar, chequeamos que esté en la opción "Codificar en UTF-8"

## Constantes matemáticas

Directorio 008

```
<!DOCTYPE HTML><html><head><script>
//Librería matemática. Constantes.
document.write(Math.E + "<br>"); //Número e
document.write(Math.LN2 + "<br>"); //Logaritmo natural de 2
document.write(Math.LN10 + "<br>"); //Logaritmo natural de 10
document.write(Math.LOG2E + "<br>"); //Logaritmo en base 2 del número de Euler
document.write(Math.LOG10E + "<br>"); //Logaritmo natural de 10 de la constante de Euler
document.write(Math.PI + "<br>"); //Constante PI
document.write(Math.SQRT1_2 + "<br>"); //Raíz cuadrada de 1/2
document.write(Math.SQRT2 + "<br>"); //Raíz cuadrada de 2
</script></head></html>
```



2.718281828459045  
0.6931471805599453  
2.302585092994046  
1.4426950408889634  
0.4342944819032518  
3.141592653589793  
0.7071067811865476  
1.4142135623730951

Ilustración 30: Constantes matemáticas

## Funciones trigonométricas

Las funciones trigonométricas deben ser con ángulos en radianes, no en grados.

Directorio 009

```
<!DOCTYPE HTML><html><head><script>
//Librería matemática.
var angGrados = 35;
document.write("Grados: " + angGrados);
var angRadian = angGrados*Math.PI/180;
document.write("<br>Radianes: " + angRadian);
var seno = Math.sin(angRadian); //Seno
document.write("<br>Seno: " + seno);
var coseno = Math.cos(angRadian); //Coseno
document.write("<br>Coseno: " + coseno);
var tangente = Math.tan(angRadian); //Tangente
document.write("<br>Tangente: " + tangente);
var arcoseno = Math.asin(seno); //Arcoseno
document.write("<br>Arcoseno: " + arcoseno);
var arcocoseno = Math.acos(coseno); //Arcocoseno
document.write("<br>Arcocoseno: " + arcocoseno);
var arctangente = Math.atan(tangente); //Arctangente
document.write("<br>Arctangente: " + arctangente);
</script></head></html>
```

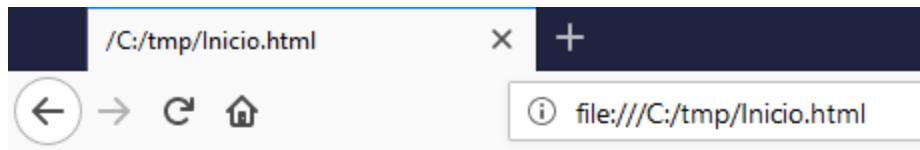
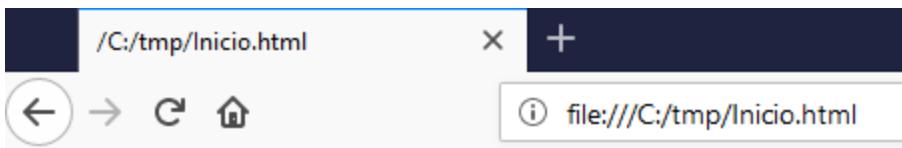


Ilustración 31: Funciones trigonométricas

# Funciones Matemáticas

Directorio 010

```
<!DOCTYPE HTML><html><head><script>
//Librería matemática.
document.write("Raíz cúbica: " + Math.cbrt(1331));
document.write("<br>Techo: " + Math.ceil(4.01));
document.write("<br>Exponencial: " + Math.exp(1)); //e elevado a N
document.write("<br>Piso: " + Math.floor(4.99));
document.write("<br>Log: " + Math.log(2)); //Logaritmo natural
document.write("<br>Mayor: " + Math.max(5, 9, 7, 3)); //Máximo valor de
una lista
document.write("<br>Menor: " + Math.min(5, 9, 7, 3)); //Mínimo valor de
una lista
document.write("<br>Potencia: " + Math.pow(2, 5)); //base, potencia
document.write("<br>Aleatorio: " + Math.random()); //Aleatorio entre 0 y 1
document.write("<br>Redondeo: " + Math.round(3.7));
document.write("<br>Redondeo: " + Math.round(3.3));
document.write("<br>Redondeo: " + Math.round(3.5));
document.write("<br>Raíz Cuadrada: " + Math.sqrt(144));
document.write("<br>Truncar: " + Math.trunc(8.923)); //Trunca al entero
</script></head></html>
```



Raíz cúbica: 11

Techo: 5

Exponencial: 2.7182818284590455

Piso: 4

Log: 0.6931471805599453

Mayor: 9

Menor: 3

Potencia: 32

Aleatorio: 0.26238171616841743

Redondeo: 4

Redondeo: 3

Redondeo: 4

Raíz Cuadrada: 12

Truncar: 8

*Ilustración 32: Funciones matemáticas*

## Captura de datos tipo texto por pantalla

Directorio 011

```
<!DOCTYPE HTML><html><head><script>
//Captura de datos por pantalla
var texto = prompt("Escriba un mensaje");
document.write("Escribió: " + texto);
</script></head></html>
```

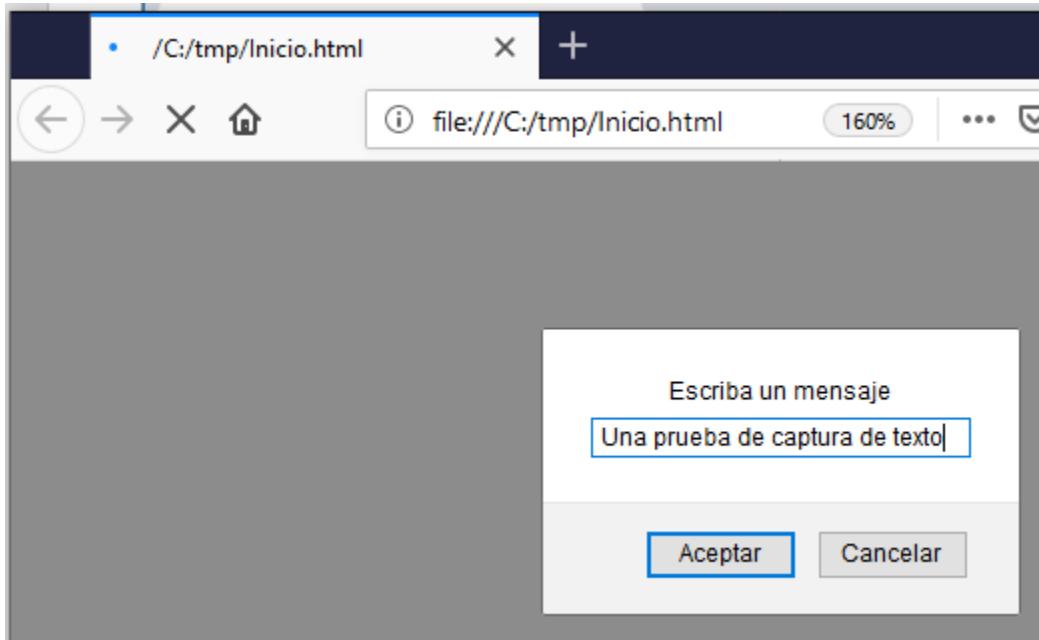


Ilustración 33: Cuadro de diálogo para capturar un valor texto



Ilustración 34: Muestra el valor capturado

## Captura de datos tipo numérico por pantalla

La instrucción “prompt” captura datos tipo texto (string), luego hay que convertirlo a un valor entero para poder hacer operaciones matemáticas. “parseInt” es la instrucción para convertir de texto a número entero y “parseFloat” es la instrucción para convertir de texto a número real.

Directorio 012

```
<!DOCTYPE HTML><html><head><script>
//Captura de datos numéricos
var numEntero = parseInt(prompt("Escriba un valor entero")); //Tipo entero
var numReal = parseFloat(prompt("Escriba un valor real")); //Tipo real
var suma = numEntero + numReal;
document.write(suma);
</script></head></html>
```

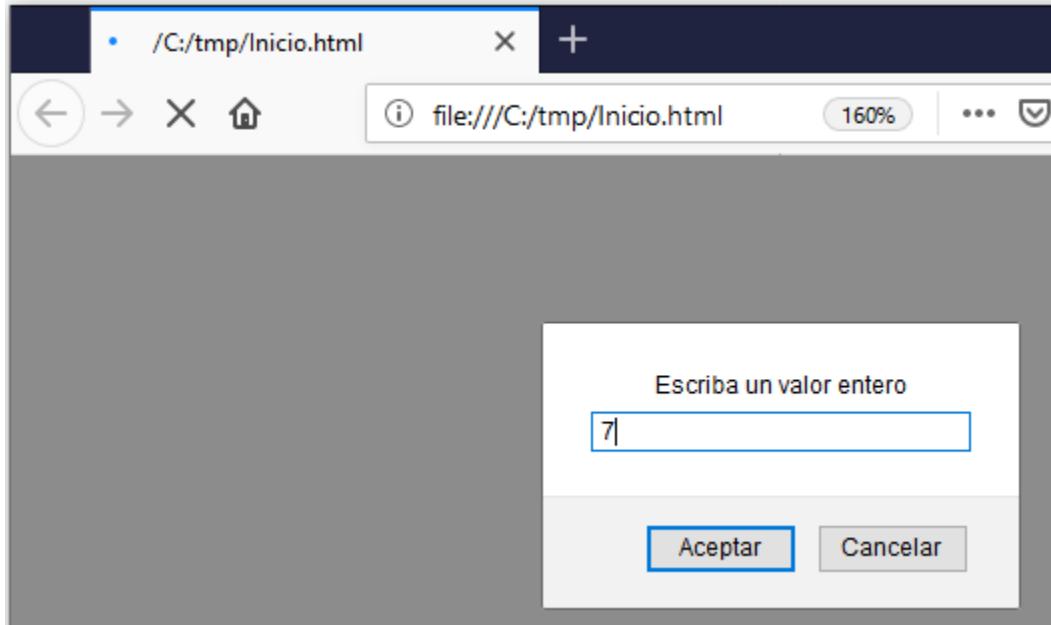


Ilustración 35: Cuadro de diálogo que captura un valor texto

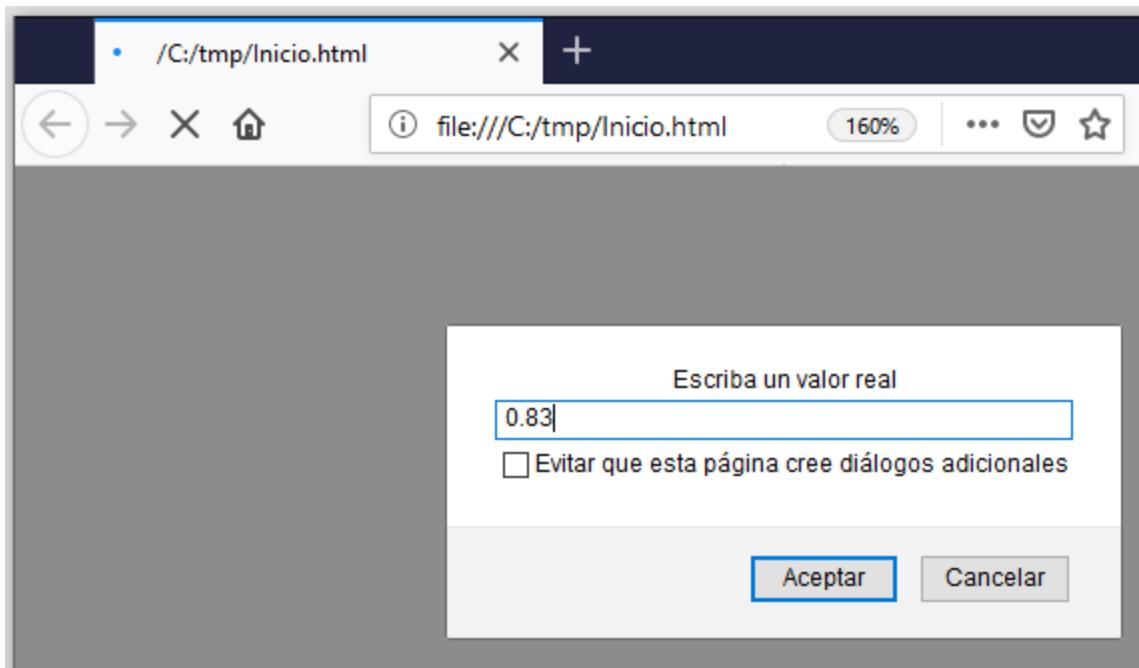


Ilustración 36: Capturando un valor texto que después se convertirá en real

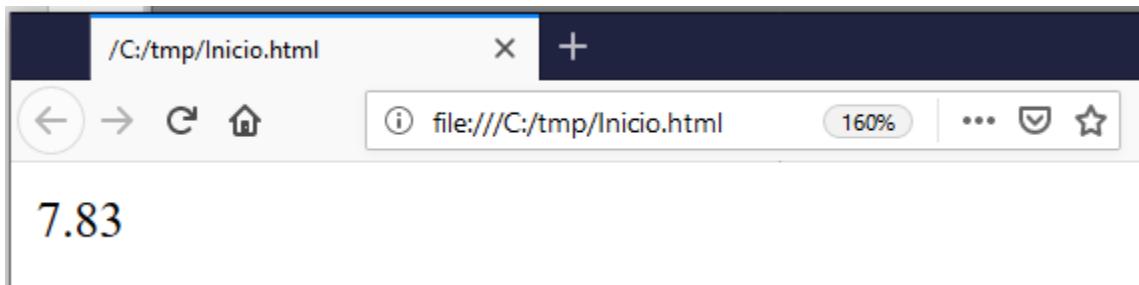


Ilustración 37: Resultado de la operación matemática

## Ejemplos de cálculos

### Cálculo del área de un triángulo dada la longitud de sus lados

Directorio 013

```
<!DOCTYPE HTML><html><head><script>
//Cálculo del área de un triángulo según sus lados
var ladoA = parseFloat(prompt("Longitud Lado A"));
var ladoB = parseFloat(prompt("Longitud Lado B"));
var ladoC = parseFloat(prompt("Longitud Lado C"));
var s = (ladoA+ladoB+ladoC)/2;
var area = Math.sqrt(s*(s-ladoA)*(s-ladoB)*(s-ladoC));
document.write("Área es: " + area);
</script></head></html>
```

### Área y perímetro del círculo

Directorio 013

```
<!DOCTYPE HTML><html><head><script>
//Área y perímetro del círculo
var radio = parseFloat(prompt("Radio"));
var area = Math.PI * radio * radio;
var perimetro = Math.PI * radio * 2;
document.write("Área es: " + area);
document.write("<br>Perímetro es: " + perimetro);
</script></head></html>
```

### Área y volumen del cilindro

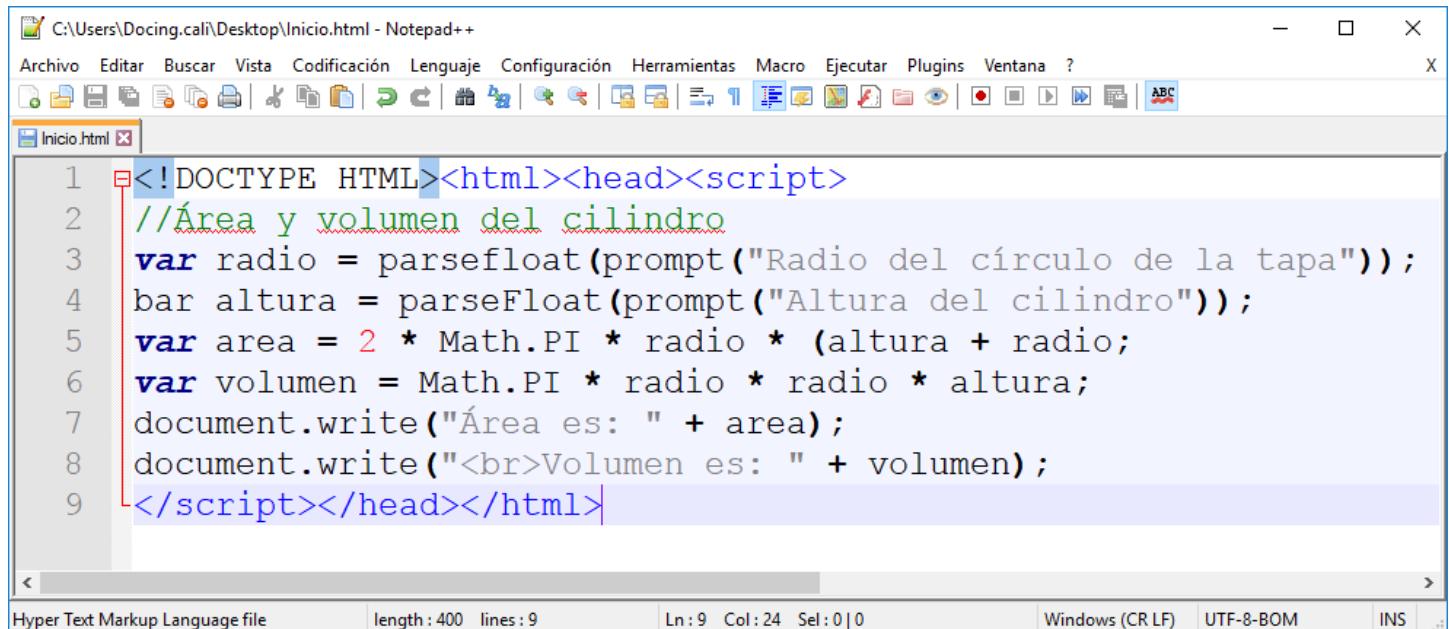
Directorio 013

```
<!DOCTYPE HTML><html><head><script>
//Área y volumen del cilindro
var radio = parseFloat(prompt("Radio del círculo de la tapa"));
var altura = parseFloat(prompt("Altura del cilindro"));
var area = 2 * Math.PI * radio * (altura + radio);
var volumen = Math.PI * radio * radio * altura;
document.write("Área es: " + area);
document.write("<br>Volumen es: " + volumen);
</script></head></html>
```

## Corrigiendo errores en JavaScript

Los navegadores modernos traen herramientas para desarrolladores que nos permiten, por ejemplo, detectar errores en nuestro código.

Tenemos el siguiente código con algunos errores:



The screenshot shows a Notepad+ window with the file 'Inicio.html' open. The code contains several syntax errors, which are underlined with red and have small red boxes with a red 'X' icon placed over them. The code is as follows:

```
1 <!DOCTYPE HTML><html><head><script>
2 //Área y volumen del cilindro
3 var radio = parseFloat(prompt("Radio del círculo de la tapa"));
4 bar altura = parseFloat(prompt("Altura del cilindro"));
5 var area = 2 * Math.PI * radio * (altura + radio);
6 var volumen = Math.PI * radio * radio * altura;
7 document.write("Área es: " + area);
8 document.write("<br>Volumen es: " + volumen);
9 </script></head></html>
```

The status bar at the bottom of the Notepad+ window displays: 'Hyper Text Markup Language file', 'length : 400 lines : 9', 'Ln: 9 Col: 24 Sel: 0 | 0', 'Windows (CR LF)', 'UTF-8-BOM', and 'INS'.

Ilustración 38: Código en JavaScript con varios errores

Al ejecutar en Mozilla Firefox, la pantalla aparece en blanco

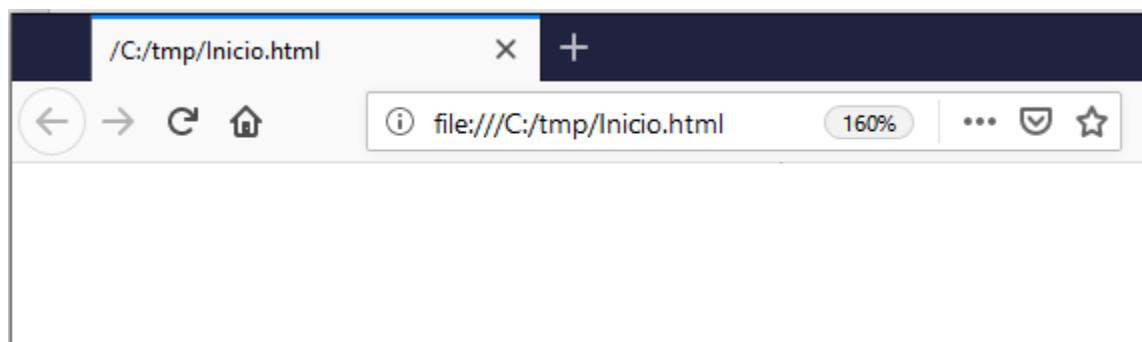


Ilustración 39: Ante errores en JavaScript, el navegador queda en blanco

En ese caso, se presiona la tecla F12 y aparece un menú de desarrollador, se selecciona “Console”

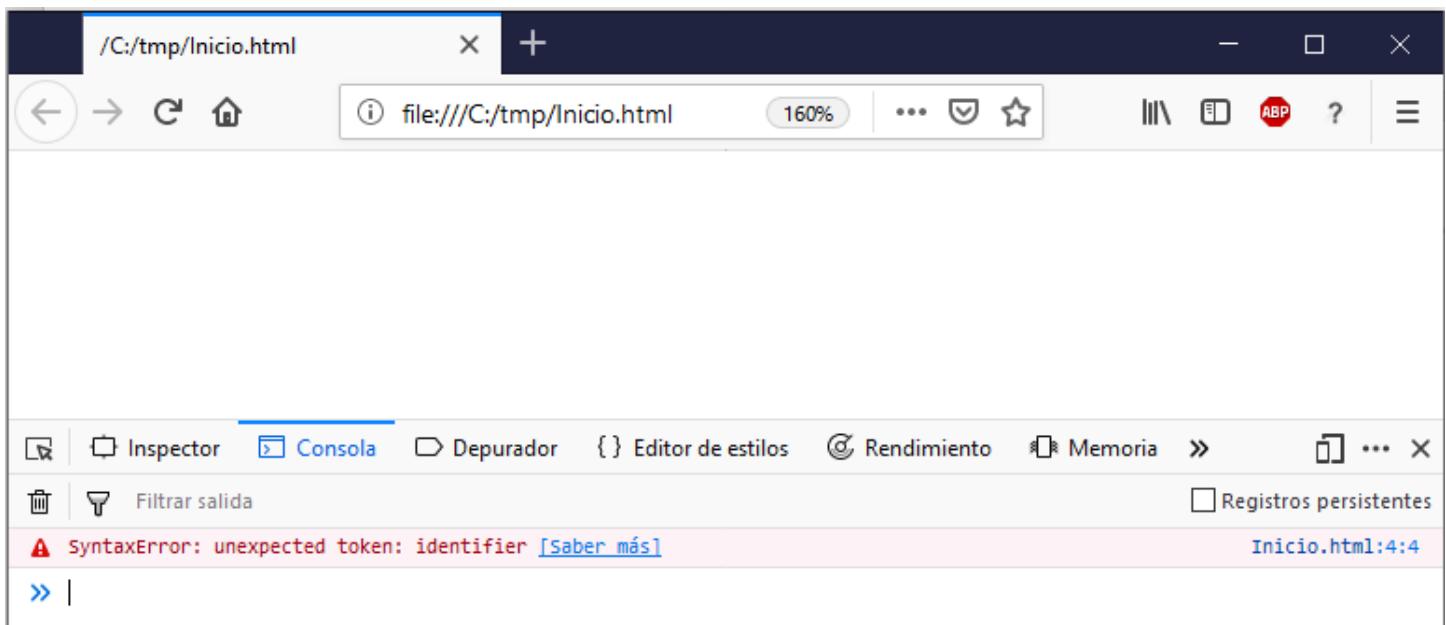


Ilustración 40: Presionando F12 en el navegador Mozilla Firefox y dando clic en Consola se muestra el error

En “Consola” se nos muestra en qué línea hay un error. Es nuestro deber entender el error y proceder a corregirlo. No debemos esperar una identificación perfecta del error por parte del navegador.

En Microsoft Edge y Google Chrome es igual, se presiona F12.

## Herramienta de Google para programar en JavaScript

Closure Compiler, dirección: <https://closure-compiler.appspot.com/home> es una herramienta muy interesante para programar en JavaScript. Esta herramienta chequea la sintaxis del código y optimiza el código.

The screenshot shows the Closure Compiler Service interface. On the left, there's a code editor containing a script to calculate the area and volume of a cylinder. The script uses Spanish variables like 'radio' and 'altura'. On the right, the results of the compilation are displayed. It shows the original size of the file (202 bytes gzipped) and the compiled size (180 bytes gzipped), indicating a 10.89% reduction. The compiled code is shown in a tabbed panel, and the output shows the correctly handled Spanish characters.

Add a URL:  Example: http://www.example.com/bigfile.js

Optimization:  Whitespace only  Simple  Advanced  
[Which optimization is right for my code?](#)

Formatting:  Pretty print  Print input delimiter

**Compile** **Reset**

```
// ==ClosureCompiler==  
// @output_file_name default.js  
// @compilation_level SIMPLE_OPTIMIZATIONS  
// ==ClosureCompiler==  
  
//Área y volumen del cilindro  
var radio = parseFloat(prompt("Radio del círculo de la tapa"));  
var altura = parseFloat(prompt("Altura del cilindro"));  
var area = 2 * Math.PI * radio * (altura + radio);  
var volumen = Math.PI * radio * radio * altura;  
document.write("Área es: " + area);  
document.write("<br>Volumen es: " + volumen);
```

Original Size: 202 bytes gzipped (333 bytes uncompressed)  
Compiled Size: 180 bytes gzipped (267 bytes uncompressed)  
Saved 10.89% off the gzipped size (19.82% without gzip)  
The code may also be accessed at [default.js](#).

Compiled Code	Warnings	Errors
POST data		
<pre>var radio=parseFloat(prompt("Radio del círculo de la tapa")),altura=parseFloat(prompt("Altura del cilindro")),area=2*Math.PI*radio*(altura+radio),volumen=Math.PI*radio*radio*altura;document.write("\u00c1rea es: "+area);document.write("&lt;br&gt;Volumen es: "+volumen);</pre>		

©2009 Google - [Terms of Service](#) - [Privacy Policy](#) - [Google Home](#)

Ilustración 41: Closure Compiler para optimizar código en JavaScript

Llama la atención el tratamiento que se le da a las tildes y caracteres propios del idioma español.

## Validación en línea del código

En <https://codebeautify.org/jsvalidate> se puede validar nuestro código de JavaScript

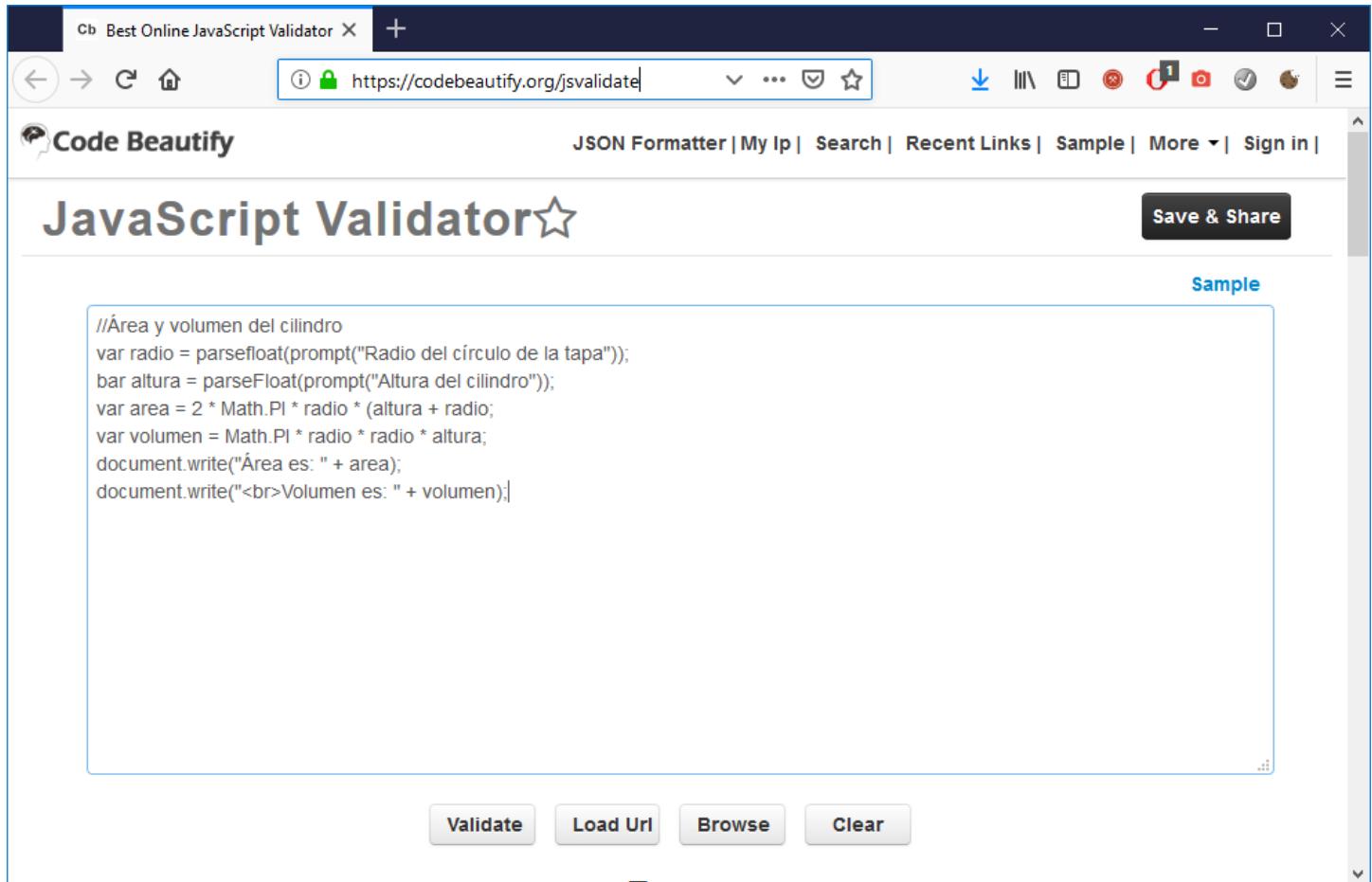


Ilustración 42: JavaScript Validator para chequear sintaxis

The screenshot shows the 'Errors' section of the Code Beautify website. It displays a table of errors with columns for 'No.', 'Line', 'Character', and 'Description'. The errors are:

No.	Line	Character	Description
1	2	13	'parseFloat' was used before it was defined. var radio = parseFloat(prompt("Radio del círculo de la tapa"));
2	2	24	'prompt' was used before it was defined. var radio = parseFloat(prompt("Radio del círculo de la tapa"));
3	3	1	'bar' was used before it was defined. bar altura = parseFloat(prompt("Altura del cilindro"));
4	3	4	Expected ';' and instead saw 'altura'. bar altura = parseFloat(prompt("Altura del cilindro"));
5	3	5	Expected 'altura' at column 1, not column 5. bar altura = parseFloat(prompt("Altura del cilindro"));
6	3	25	'prompt' was used before it was defined. bar altura = parseFloat(prompt("Altura del cilindro"));
7	4	35	'altura' was used before it was defined. var area = 3.14 * radio * radio * altura;

Ilustración 43: Diagnóstico de JavaScript Validator de un script de ejemplo

Otro sitio es <https://www.webtoolkitonline.com/javascript-tester.html>

The screenshot shows the 'JavaScript Tester' section of the Web Toolkit Online website. On the left, there's a sidebar with links for 'CONVERTER', 'ENCODER / DECODER', and 'FORMATTER'. The main area shows the 'JavaScript Tester' interface with a 'JavaScript code' editor. The code is:

```

1 //Área y volumen del cilindro
2 var radio = parseFloat(prompt("Radio del círculo de la tapa"));
3 bar altura = parseFloat(prompt("Altura del cilindro"));
4 Missing ";" before statement|dio * (altura + radio;
5 |radio * radio * altura;
6 |document.write("Área es: " + area);
7 |document.write("<br>Volumen es: " + volumen);

```

The line 'bar altura =' has two red error markers, indicating a variable declaration error. Lines 4 and 5 have yellow warning markers, indicating a missing semicolon and a syntax error respectively.

Ilustración 44:JavaScript Tester para validar sintaxis

Y un tercero, es en <https://jshint.com/>

The screenshot shows a browser window with the title "JSHint, a JavaScript Code Quality Tool". The address bar displays "https://jshint.com". The main content area shows a block of JavaScript code and its analysis results:

```
1 //Área y volumen del cilindro
2 var radio = parseFloat(prompt("Radio del círculo de la tapa"));
3 var altura = parseFloat(prompt("Altura del cilindro"));
4 var area = 2 * Math.PI * radio * (altura + radio);
5 var volumen = Math.PI * radio * radio * altura;
6 document.write("Área es: " + area);
7 document.write("<br>Volumen es: " + volumen);
```

**CONFIGURE**

Six warnings

- 3 Expected an assignment or function call and instead saw an expression.
- 3 Missing semicolon.
- 4 Expected ')' to match '(' from line 4 and instead saw '}'.
- 4 Missing semicolon.
- 6 document.write can be a form of eval.
- 7 document.write can be a form of eval.

Three undefined variables

- 2 parseFloat
- 3 bar
- 3 altura
- 4 altura
- 5 altura

The right sidebar of the browser shows the JS Hint logo and version 2.9.6, with links to "About", "Documentation", "Install", "Contribute", and "Blog".

Ilustración 45: Jshint para validar sintaxis

## Sí condicional

Similar a C++ o C# es la sintaxis del sí condicional en JavaScript.

Directorio 014

```
<!DOCTYPE HTML><html><head><script>
//Si condicional
var valA = parseFloat(prompt("valor A: "));
var valB = parseFloat(prompt("valor B: "));
if (valA > valB){
    document.write(valA + " es mayor que " + valB);
}
else {
    document.write(valA + " es menor o igual que " + valB);
}
</script></head></html>
```

Directorio 014

```
<!DOCTYPE HTML><html><head><script>
//Si condicional. Uso de else if
var valA = parseFloat(prompt("valor A: "));
var valB = parseFloat(prompt("valor B: "));
if (valA > valB){
    document.write(valA + " es mayor que " + valB);
} else if (valA == valB){
    document.write(valA + " es igual que " + valB);
}
else {
    document.write(valA + " es menor que " + valB);
}
</script></head></html>
```

Los operadores de comparación son:

Símbolo	Significado
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que
==	Es igual a
!=	Es diferente que
==	Comparación de igualdad estricta de valor y tipo. Por ejemplo 5 == "5" sería falso En cambio 5 == "5" sería verdadero (porque == es un comparador menos estricto)
!==	Comparación de diferencia estricta de valor y tipo.

## Operadores lógicos: Conjunción (Y, &&) y Disyunción (O, ||)

El operador AND, Y en JavaScript es &&. El operador OR, O en JavaScript es ||

Directorio 015

```
<!DOCTYPE HTML><html><head><script>
//Tipo de triángulo
var ladoA = parseInt(prompt("Lado A:"));
var ladoB = parseInt(prompt("Lado B:"));
var ladoC = parseInt(prompt("Lado C:"));
if (ladoA==ladoB && ladoB==ladoC) {
    document.write("Es equilátero");
} else if (ladoA!=ladoB && ladoA!=ladoC && ladoB!=ladoC) {
    document.write("Es escaleno");
} else {
    document.write("Es isósceles");
}
</script></head></html>
```

Directorio 015

```
<!DOCTYPE HTML><html><head><script>
//Tipo de triángulo
var ladoA = parseInt(prompt("Lado A:"));
var ladoB = parseInt(prompt("Lado B:"));
var ladoC = parseInt(prompt("Lado C:"));
if (ladoA==ladoB && ladoB==ladoC) {
    document.write("Es equilátero");
} else if (ladoA==ladoB || ladoB==ladoC || ladoA==ladoC) {
    document.write("Es isósceles");
} else {
    document.write("Es escaleno");
}
</script></head></html>
```

## Uso del switch

Directorio 016

```
<!DOCTYPE HTML><html><head><script>
//Uso del switch
var valor = parseInt(prompt("Digite un valor"));
switch(valor) {
    case 0: document.write("Escribió cero"); break;
    case 1: document.write("Digitó 1"); break;
    case 2: document.write("Tecleó dos"); break;
    case 3: document.write("Presionó 3"); break;
    default: document.write("Ingresó otro valor");
}
</script></head></html>
```

Se pueden usar varias líneas en el switch. El cierre es la palabra reservada “break”.

Directorio 016

```
<!DOCTYPE HTML><html><head><script>
//Uso del switch
var valor = parseInt(prompt("Digite un valor"));
switch(valor) {
    case 0: document.write("Escribió cero"); break;
    case 1: document.write("Digitó 1");
              document.write("<br>Escribió uno");
              document.write("<br>Registró uno");
              break;
    case 2: document.write("Tecleó dos"); break;
    case 3: document.write("Presionó 3"); break;
    default: document.write("Ingresó otro valor");
}
</script></head></html>
```

## Ciclos. Uso del “for”

Directorio 017

```
<!DOCTYPE HTML><html><head><script>
//Ciclo for creciente
for (var cont = 1; cont<=100; cont++) {
    document.write(cont + "<br>");
}
</script></head></html>
```

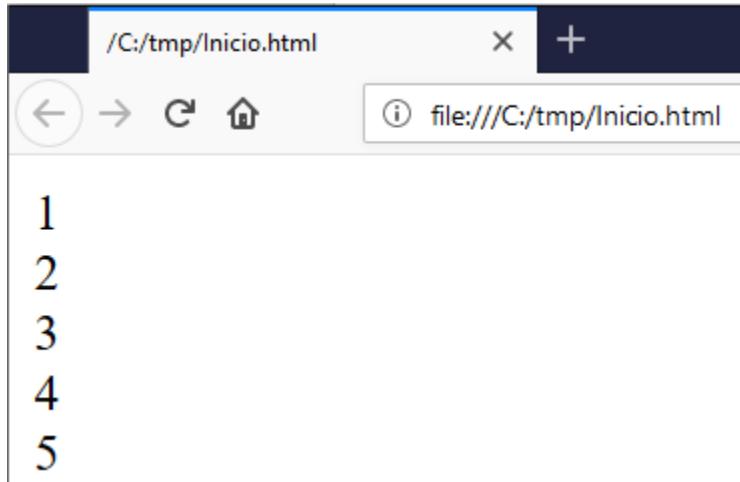


Ilustración 46: Ejecución del ciclo ascendente

Directorio 017

```
<!DOCTYPE HTML><html><head><script>
//Ciclo for decreciente
for (var cont = 100; cont>=1; cont--) {
    document.write(cont + "<br>");
}
</script></head></html>
```

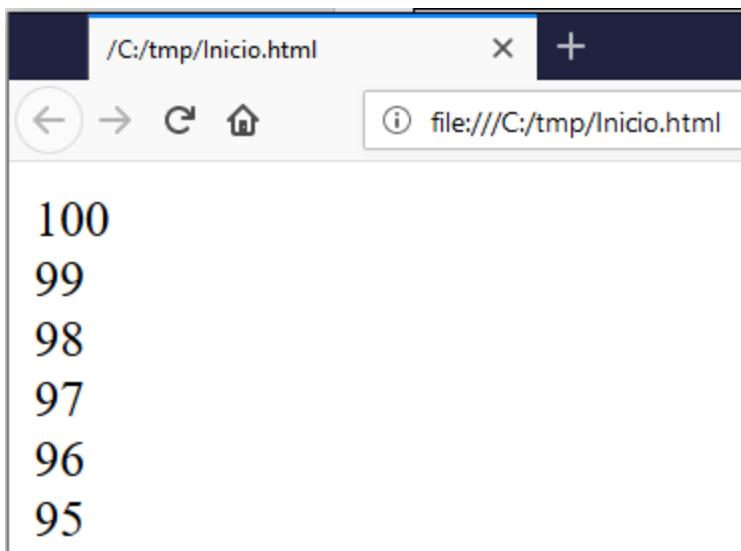


Ilustración 47: Ejecución del ciclo descendente

Directorio 017

```
<!DOCTYPE HTML><html><head><script>
//Ciclo for creciente de 3 en 3
for (var cont = 0; cont<=300; cont+=3) {
    document.write(cont + "<br>");
}
</script></head></html>
```

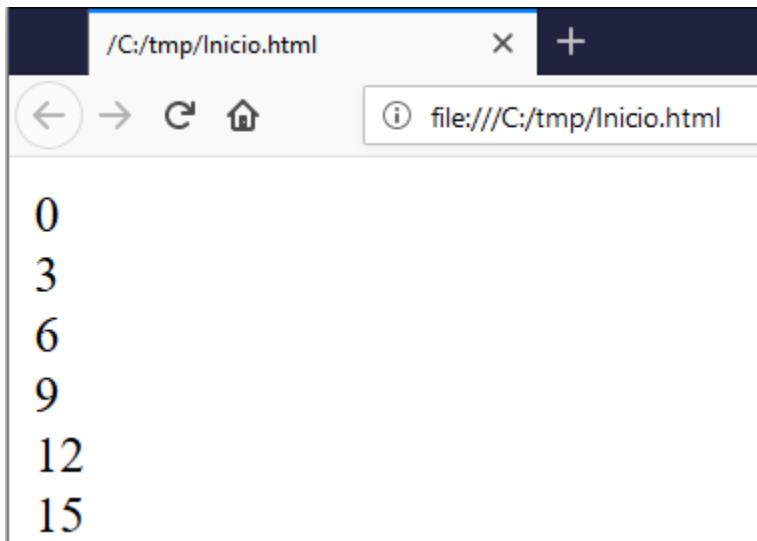


Ilustración 48: Ciclo ascendente de 3 en 3

Directorio 017

```
<!DOCTYPE HTML><html><head><script>
//Ciclo for creciente de 10 en 10. Se detiene antes de 500
for (var cont = 0; cont<500; cont+=10) {
    document.write(cont + "<br>");
}
```

```
</script></head></html>
```

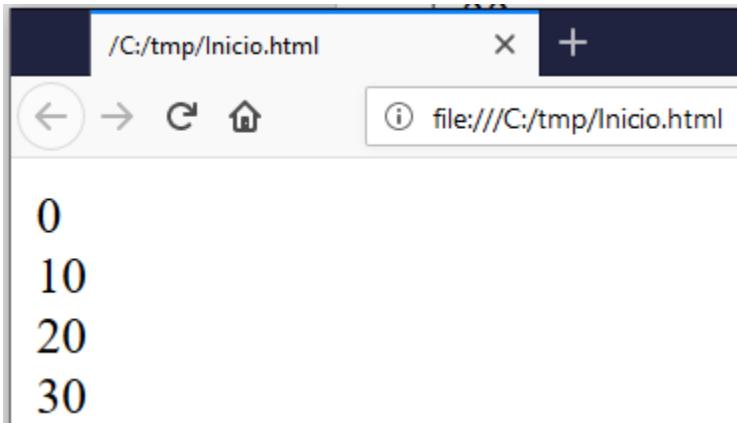


Ilustración 49: Ciclo ascendente de 10 en 10

Directorio 017

```
<!DOCTYPE HTML><html><head><script>  
//Ciclo for creciente aumentando el doble del anterior  
for (var cont = 1; cont<=5000; cont*=2){  
    document.write(cont + ", ");  
}  
//Resultado:1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096  
</script></head></html>
```



Ilustración 50: Ciclo ascendente doblando el anterior

Directorio 017

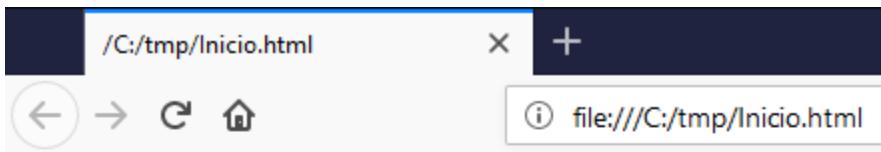
```
<!DOCTYPE HTML><html><head><script>  
//Ciclo for decreciente disminuyendo la mitad del anterior  
for (var cont = 4096; cont>=1; cont/=2){  
    document.write(cont + ", ");  
}  
//Resultado: 4096, 2048, 1024, 512, 256, 128, 64, 32, 16, 8, 4, 2, 1,  
</script></head></html>
```



Ilustración 51: Ciclo descendente dividiendo entre 2 el anterior

Directorio 017

```
<!DOCTYPE HTML><html><head><script>
/* Ciclo for con dos variables. Observemos las dos
   condiciones que deben darse */
for (var x=1, y=34; x<=10 && y>=28; x++, y--) {
    document.write(x + ", " + y + "<br>");
}
/* Resultado:
1, 34
2, 33
3, 32
4, 31
5, 30
6, 29
7, 28
*/
</script></head></html>
```



1, 34  
2, 33  
3, 32  
4, 31  
5, 30  
6, 29  
7, 28

Ilustración 52: Ciclo que muestra el progreso de dos variables

```
<!DOCTYPE HTML><html><head><script>
/* Ciclo for con dos variables. Observemos las dos
   condiciones que deben darse cambiando && por || */
for (var x=1, y=34; x<=10 || y>=28; x++, y--) {
    document.write(x + ", " + y + "<br>");
}
/* Resultado:
1, 34
2, 33
3, 32
4, 31
5, 30
6, 29
7, 28
8, 27
9, 26
10, 25
*/
</script></head></html>
```

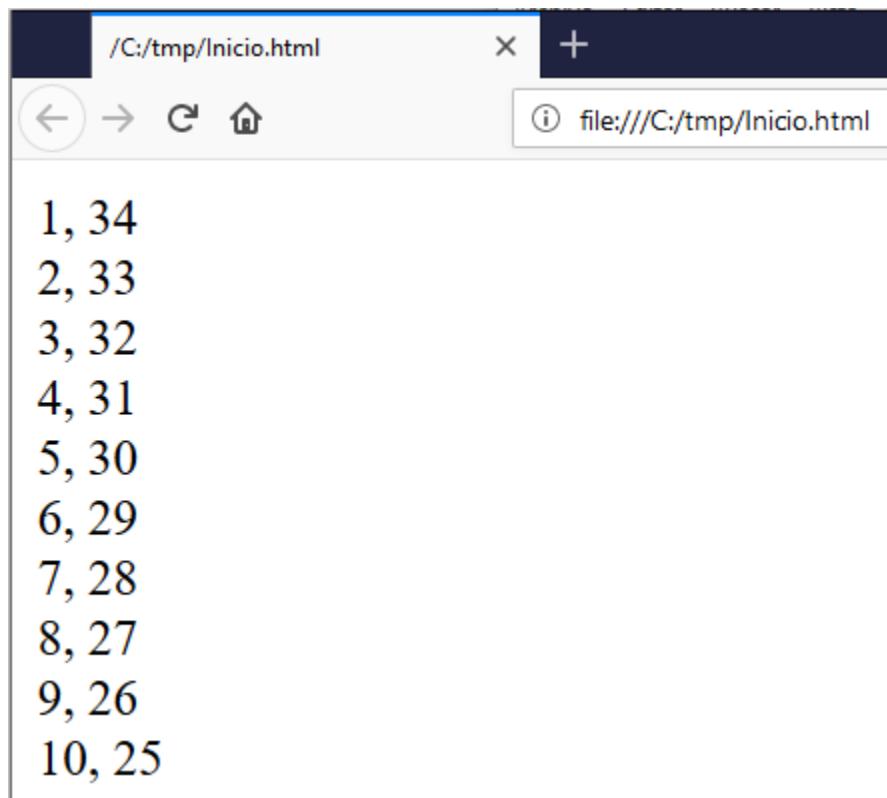


Ilustración 53: Ciclo que muestra el progreso de dos variables:

## Ciclo while

Directorio 018

```
<!DOCTYPE HTML><html><head><script>
//Ciclo while creciente
var cont = 1;
while (cont<=100) {
    document.write(cont + "<br>");
    cont++;
}
</script></head></html>
```

Directorio 018

```
<!DOCTYPE HTML><html><head><script>
//Ciclo while decreciente
var cont = 100;
while (cont>=1) {
    document.write(cont + "<br>");
    cont--;
}
</script></head></html>
```

Directorio 018

```
<!DOCTYPE HTML><html><head><script>
//Ciclo while creciente de 3 en 3
var cont = 0;
while (cont<=300) {
    document.write(cont + "<br>");
    cont+=3;
}
</script></head></html>
```

Directorio 018

```
<!DOCTYPE HTML><html><head><script>
//Ciclo while creciente de 10 en 10. Se detiene antes de 500
var cont = 0;
while (cont<500) {
    document.write(cont + "<br>");
    cont+=10;
}
</script></head></html>
```

Directorio 018

```
<!DOCTYPE HTML><html><head><script>
//Ciclo while creciente aumentando el doble del anterior
var cont = 1;
```

```

while (cont<=5000){
    document.write(cont + ", ");
    cont*=2;
}
//Resultado:1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096
</script></head></html>

```

*Directorio 018*

```

<!DOCTYPE HTML><html><head><script>
//Ciclo while decreciente disminuyendo la mitad del anterior
var cont = 4096;
while (cont>=1){
    document.write(cont + ", ");
    cont/=2;
}
//Resultado: 4096, 2048, 1024, 512, 256, 128, 64, 32, 16, 8, 4, 2, 1,
</script></head></html>

```

*Directorio 018*

```

<!DOCTYPE HTML><html><head><script>
/* Ciclo while con dos variables. Observemos las dos
   condiciones que deben darse */
var x=1;
var y=34;
while (x<=10 && y>=28){
    document.write(x + " " + y + "<br>");
    x++;
    y--;
}
/* Resultado:
1, 34
2, 33
3, 32
4, 31
5, 30
6, 29
7, 28
*/
</script></head></html>

```

*Directorio 018*

```

<!DOCTYPE HTML><html><head><script>
/* Ciclo while con dos variables. Observemos las dos
   condiciones que deben darse cambiando && por || */
var x=1;
var y=34;
while (x<=10 || y>=28){
    document.write(x + " " + y + "<br>");
}

```

```
    x++;
    y--;
}
/* Resultado:
1, 34
2, 33
3, 32
4, 31
5, 30
6, 29
7, 28
8, 27
9, 26
10, 25
*/
</script></head></html>
```

## Ciclo do - while

Directorio 019

```
<!DOCTYPE HTML><html><head><script>
//Ciclo do-while creciente
var cont = 1;
do{
    document.write(cont + "<br>");
    cont++;
}while (cont<=100);
</script></head></html>
```

Directorio 019

```
<!DOCTYPE HTML><html><head><script>
//Ciclo do-while decreciente
var cont = 100;
do{
    document.write(cont + "<br>");
    cont--;
}while (cont>=1);
</script></head></html>
```

Directorio 019

```
<!DOCTYPE HTML><html><head><script>
//Ciclo do-while creciente de 3 en 3
var cont = 0;
do{
    document.write(cont + "<br>");
    cont+=3;
}while (cont<=300);
</script></head></html>
```

Directorio 019

```
<!DOCTYPE HTML><html><head><script>
//Ciclo do-while creciente de 10 en 10. Se detiene antes de 500
var cont = 0;
do{
    document.write(cont + "<br>");
    cont+=10;
}while (cont<500);
</script></head></html>
```

Directorio 019

```

<!DOCTYPE HTML><html><head><script>
//Ciclo while creciente aumentando el doble del anterior
var cont = 1;
do{
    document.write(cont + ", ");
    cont*=2;
}while (cont<=5000);
//Resultado:1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096
</script></head></html>

```

*Directorio 019*

```

<!DOCTYPE HTML><html><head><script>
//Ciclo while decreciente disminuyendo la mitad del anterior
var cont = 4096;
do{
    document.write(cont + ", ");
    cont/=2;
}while (cont>=1);
//Resultado: 4096, 2048, 1024, 512, 256, 128, 64, 32, 16, 8, 4, 2, 1,
</script></head></html>

```

*Directorio 019*

```

<!DOCTYPE HTML><html><head><script>
/* Ciclo while con dos variables. Observemos las dos
   condiciones que deben darse */
var x=1;
var y=34;
do{
    document.write(x + " " + y + "<br>");
    x++;
    y--;
}while (x<=10 && y>=28);
/* Resultado:
1, 34
2, 33
3, 32
4, 31
5, 30
6, 29
7, 28
*/
</script></head></html>

```

*Directorio 019*

```

<!DOCTYPE HTML><html><head><script>
/* Ciclo do-while con dos variables. Observemos las dos
   condiciones que deben darse cambiando && por || */
var x=1;

```

```
var y=34;
do{
    document.write(x + " " + y + "<br>");
    x++;
    y--;
}while (x<=10 || y>=28);
/* Resultado:
1, 34
2, 33
3, 32
4, 31
5, 30
6, 29
7, 28
8, 27
9, 26
10, 25
*/
</script></head></html>
```

## Uso del break

Directorio 020

```
<!DOCTYPE HTML><html><head><script>
//Ciclo for creciente, se detiene en 50
for (var cont = 1; cont<=100; cont++) {
    document.write(cont + "<br>");
    if (cont>=50) break; //Sale del ciclo
}
</script></head></html>
```

Directorio 020

```
<!DOCTYPE HTML><html><head><script>
//Ciclo while creciente. Sale cuando cont sea 50
var cont = 1;
while (cont<=100){
    document.write(cont + "<br>");
    cont++;
    if (cont>=50) break; //Sale del ciclo
}
</script></head></html>
```

Directorio 020

```
<!DOCTYPE HTML><html><head><script>
//Ciclo do-while creciente. Sale cuando cont sea 50
var cont = 1;
do{
    document.write(cont + "<br>");
    cont++;
    if (cont>=50) break; //Sale del ciclo
}while (cont<=100);
</script></head></html>
```

## Uso del continue

La sentencia “continue” hace que el programa haga inmediatamente la siguiente iteración ignorando las instrucciones siguientes dentro del ciclo.

*Directorio 021*

```
<!DOCTYPE HTML><html><head><script>
//Ciclo for creciente, se salta los pares
for (var cont = 1; cont<=10; cont++) {
    //Si es par, salta a la siguiente iteración
    if (cont%2==0) continue;
    document.write(cont + ", ");
}
//Resultado: 1, 3, 5, 7, 9,
</script></head></html>
```

*Directorio 021*

```
<!DOCTYPE HTML><html><head><script>
//Ciclo while creciente. Se salta los pares
var cont = 1;
while (cont<=100) {
    cont++;
    if (cont%2==0) continue; //Salta a la siguiente iteración
    document.write(cont + "<br>");
}
</script></head></html>
```

*Directorio 021*

```
<!DOCTYPE HTML><html><head><script>
//Ciclo do-while creciente. Se salta los pares
var cont = 1;
do{
    cont++;
    if (cont%2==0) continue; //Salta a la siguiente iteración
    document.write(cont + "<br>");
}while (cont<=100);
</script></head></html>
```

## Ciclos anidados y salir de estos

Directorio 022

```
<!DOCTYPE HTML><html><head><script>
//Ciclos anidados
for (var x = 1; x <= 3; x++) {
    for (var y = 45; y <= 50; y++) {
        for (var z = 7; z <= 9; z++) {
            document.write(x + ";" + y + ";" + z + "<br>");
        }
    }
}
</script></head></html>
```

Para salir de ciclos anidados hacemos uso de etiquetas, luego hacemos uso de “break etiqueta” y saltaría al lugar de esa etiqueta:

Directorio 022

```
<!DOCTYPE HTML><html><head><script>
//Saliendo de ciclos anidados. Uso de etiquetas.
saliendo:
for (var x = 1; x <= 3; x++) {
    for (var y = 45; y <= 50; y++) {
        for (var z = 7; z <= 9; z++) {
            document.write(x + ";" + y + ";" + z + "<br>");
            if (z==8) break saliendo; //Sale de todos los ciclos
        }
    }
}
/* Resultado:
1;45;7
1;45;8 */
</script></head></html>
```

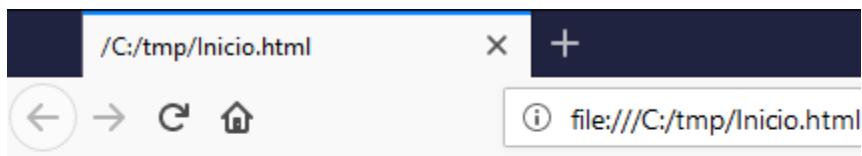
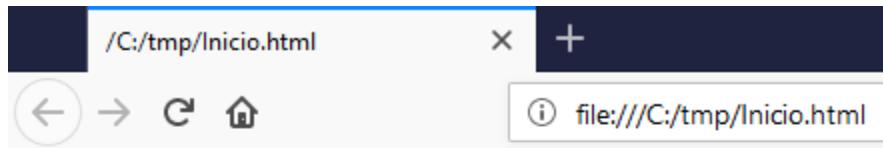


Ilustración 54: Ciclos anidados y salir de estos

```
<!DOCTYPE HTML><html><head><script>
//Saliendo de ciclos anidados
for (var x = 1; x <= 3; x++) {
    ciclointerno:
    for (var y = 45; y <= 50; y++) {
        for (var z = 7; z <= 9; z++) {
            document.write(x + ";" + y + ";" + z + "<br>");
            if (z==8) break ciclointerno;
        }
    }
}
/*Resultado:
1;45;7
1;45;8
2;45;7
2;45;8
3;45;7
3;45;8 */
</script></head></html>
```



1;45;7  
1;45;8  
2;45;7  
2;45;8  
3;45;7  
3;45;8

Ilustración 55: Ciclos anidados y salir de estos

## Funciones o subrutinas

Directorio 023

```
<!DOCTYPE HTML><html><head><script>
var numEntero = parseInt(prompt("Escriba un valor entero")); //Tipo entero
var Primo = EsPrimo(numEntero); //Llama la función
document.write(numEntero + " es primo: " + Primo);

//Función que retorna true si el número enviado por parámetro es primo
function EsPrimo(num){
    if (num<=1) return false;
    if (num==2) return true;
    if (num%2==0) return false;
    for (var divide = 3; divide <= Math.sqrt(num); divide+=2)
        if (num%divide == 0) return false;
    return true;
}
</script></head></html>
```

El anterior programa pregunta un número por pantalla y responde si el número es primo o no.

Cuando tenemos una buena cantidad de funciones, es recomendable crear una librería. Generamos un archivo con extensión .js

```
C:\tmp\Libreria.js - Notepad++
Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Plugins Ventana ?
Libreria.js x
1 //Retorna true si el número es palíndromo
2 function EsPalindromo(num) {
3     if (num==InvierteNumero(num)) return true;
4     return false;
5 }
6
7 //Retorna las dos últimas cifras del número
8 function retornadosultimascifras(num) {
9     return num%100;
10}
11
```

Ilustración 56: Funciones, subrutinas o procedimientos

Y este sería el código dentro de ese archivo Libreria.js

```

//Retorna true si el número es palíndromo
function EsPalindromo(num) {
    if (num==InvierteNumero(num)) return true;
    return false;
}

//Retorna las dos últimas cifras del número
function retornadosultimascifras(num) {
    return num%100;
}

//Dice cuántas cifras de determinado número hay en el número enviado
function cifrashalladas(num, cifra){
    var acumula = 0;
    while (num!=0){
        if ( num%10 == cifra) acumula++;
        num = Math.floor(num/10);
    }
    return acumula;
}

//Invierte un número
function InvierteNumero(num) {
    var multiplica = Math.pow(10, TotalCifras(num)-1);
    var acumula = 0;
    while(num!=0){
        cifra = num%10;
        acumula += cifra * multiplica;
        multiplica /= 10;
        num = Math.floor(num/10);
    }
    return acumula;
}

//Retorna la primera cifra de un número
function PrimeraCifra(num){
    var primera = 0;
    while(num!=0){
        primera = num%10;
        num = Math.floor(num/10);
    }
    return primera;
}

//Retorna el número de cifras de un número
function TotalCifras(num){
    var cuenta = 0;
    while(num!=0){
        cuenta++;
}

```

```

        num = Math.floor(num/10);
    }
    return cuenta;
}

//Retorna true si un número es impar
function EsImpar(num){
    if (num%2==1) return true;
    return false;
}

//Retorna true si un número es par
function EsPar(num){
    if (num%2==0) return true;
    return false;
}

//Retorna la sumatoria de las cifras de un número
function SumaCifras(num){
    var acum = 0;
    while(num!=0){
        var cifra = num%10;
        acum += cifra;
        num = Math.floor(num/10);
    }
    return acum;
}

//Retorna la sumatoria de las cifras pares de un número
function SumaCifrasPares(num){
    var acum = 0;
    while(num!=0){
        var cifra = num%10;
        if (cifra%2==0)      acum += cifra;
        num = Math.floor(num/10);
    }
    return acum;
}

//Retorna la sumatoria de las cifras impares de un número
function SumaCifrasImpares(num){
    var acum = 0;
    while(num!=0){
        var cifra = num%10;
        if (cifra%2!=0)      acum += cifra;
        num = Math.floor(num/10);
    }
    return acum;
}

//Retorna el producto de las cifras de un número

```

```

function MultiplicaCifras(num) {
    var acum = 1;
    while(num!=0){
        var cifra = num%10;
        acum *= cifra;
        num = Math.floor(num/10);
    }
    return acum;
}

//Retorna el producto de las cifras impares de un número
function MultiplicaCifrasImpares(num){
    var acum = 1;
    while(num!=0){
        var cifra = num%10;
        if (cifra%2!=0) acum *= cifra;
        num = Math.floor(num/10);
    }
    return acum;
}

//Retorna el producto de las cifras pares de un número
function MultiplicaCifrasPares(num){
    var acum = 1;
    while(num!=0){
        var cifra = num%10;
        if (cifra%2==0) acum *= cifra;
        num = Math.floor(num/10);
    }
    return acum;
}

//Retorna la antepenúltima cifra de un número entero
function AntepenultimaCifra(num){
    return Math.floor(num/100)%10;
}

//Retorna la penúltima cifra de un número entero
function PenultimaCifra(num){
    return Math.floor(num/10)%10;
}

//Retorna la última cifra de un número entero
function UltimaCifra(num){
    return num%10;
}

//Retorna true si el número enviado por parámetro es primo
function EsPrimo(num){
    if (num<=1) return false;
    if (num==2) return true;
}

```

```

if (num%2==0) return false;
for (var divide = 3; divide <= Math.sqrt(num); divide+=2)
    if (num%divide == 0) return false;
return true;
}

//Retorna true si todas las cifras son pares
function TodasCifrasPares(num){
    while(num!=0){
        var cifra = num%10;
        if (cifra%2!=0) return false;
        num = Math.floor(num/10);
    }
    return true;
}

//Retorna true si todas las cifras son impares
function TodasCifrasImpares(num){
    if (num==0) return false;
    while(num!=0){
        var cifra = num%10;
        if (cifra%2==0) return false;
        num = Math.floor(num/10);
    }
    return true;
}

//Retorna el número de cifras pares
function TotalCifrasPares(num){
    var acum = 0;
    while(num!=0){
        var cifra = num%10;
        if (cifra%2==0) acum++;
        num = Math.floor(num/10);
    }
    return acum;
}

//Retorna el número de cifras impares
function TotalCifrasImpares(num){
    var acum = 0;
    while(num!=0){
        var cifra = num%10;
        if (cifra%2!=0) acum++;
        num = Math.floor(num/10);
    }
    return acum;
}

//Retorna la cifra más alta
function LaCifraMasAlta(num){

```

```

var cifra = 0;
    while (num!=0){
        if ( num%10 > cifra) cifra = num%10;
        num = Math.floor(num/10);
    }
    return cifra;
}

//Retorna la cifra más baja
function LaCifraMasBaja(num){
    var cifra = 9;
    while (num!=0){
        if ( num%10 < cifra) cifra = num%10;
        num = Math.floor(num/10);
    }
    return cifra;
}

//Retorna true si num tiene sólo cifras menores o iguales de cifra
function SoloCifrasMenorIgual(num, cifra){
    while (num!=0){
        if ( num%10 > cifra) return false;
        num = Math.floor(num/10);
    }
    return true;
}

//Retorna true si num tiene sólo cifras mayores o iguales de cifra
function SoloCifrasMayorIgual(num, cifra){
    while (num!=0){
        if ( num%10 < cifra) return false;
        num = Math.floor(num/10);
    }
    return true;
}

//Retorna true si todas las cifras son distintas
function DistintasCifras(num){
    for (var cifra=0; cifra<=9; cifra++){
        var numero = num;
        var cuenta = 0;
        while (numero!=0){
            if (numero%10 == cifra) cuenta++;
            if (cuenta > 1) return false;
            numero = Math.floor(numero/10);
        }
    }
    return true;
}

//Retorna si todas las cifras pares son distintas

```

```

function DistintasCifrasPares(num) {
    for (var cifra=0; cifra<=8; cifra+=2){
        var numero = num;
        var cuenta = 0;
        while (numero!=0){
            if (numero%10 == cifra) cuenta++;
            if (cuenta > 1) return false;
            numero = Math.floor(numero/10);
        }
    }
    return true;
}

//Retorna si todas las cifras impares son distintas
function DistintasCifrasImpares(num){
    for (var cifra=1; cifra<=9; cifra+=2){
        var numero = num;
        var cuenta = 0;
        while (numero!=0){
            if (numero%10 == cifra) cuenta++;
            if (cuenta > 1) return false;
            numero = Math.floor(numero/10);
        }
    }
    return true;
}

//Elevar al cuadrado cada cifra y retornar la suma
function sumacifrascuadrado(num){
    var acumula = 0;
    while (num!=0){
        acumula+= (num%10)*(num%10);
        num = Math.floor(num/10);
    }
    return acumula;
}

```

Para usar la librería se llama desde el archivo principal con <script src="Librería.js"></script>

### Directorio 023

```

<!DOCTYPE HTML><html><head><script src="Libreria.js"></script>
<script>
var numEntero = parseInt(prompt("Escribir un valor entero"));
var Primo = EsPrimo(numEntero); //Llama la función
document.write(numEntero + " es primo: " + Primo);
</script></head></html>

```

De esa forma separa el código JavaScript y hace más legible nuestro código.

## Ejemplos de uso de la librería

Directorio 023

```
<!DOCTYPE HTML><html><head>
<script src="Libreria.js"></script>
<script>
var minimo = 1000;
var maximo = 9999;

//Cantidad de números que la suma de las tres últimas cifras es par
var cuenta = 0;
for (var num = minimo; num<=maximo; num++) {
    if (EsPar(AntepenultimaCifra(num) + UltimaCifra(num) +
PenultimaCifra(num))) {
        cuenta++;
    }
}
Imprime(minimo, maximo, "Cantidad de números que la suma de las tres
últimas cifras es par", cuenta);

//Imprime el resultado en el navegador
function Imprime(minimo, maximo, texto, resultado){
    document.write("Entre " + minimo + " y " + maximo + ". " + texto + ":" +
" " + resultado + "<br>");
}
</script></head></html>
```

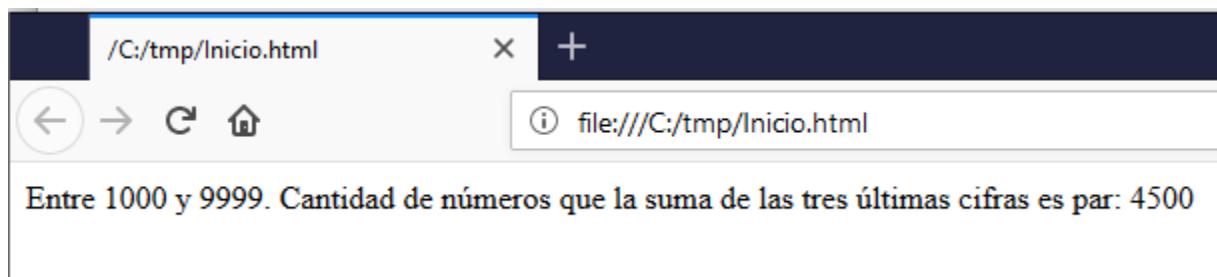


Ilustración 57: Entre 1000 y 9999. Cantidad de números que la suma de las tres últimas cifras es par: 4500

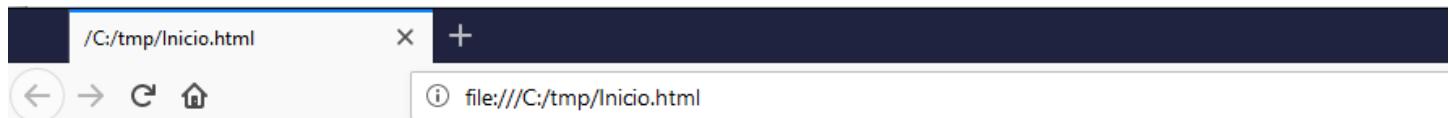
```

<!DOCTYPE HTML><html><head>
<script src="Libreria.js"></script>
<script>
var minimo = 1000;
var maximo = 9999;

//Cantidad de números que la multiplicación de las tres últimas cifras es
igual a la suma de esas tres últimas cifras
var cuenta = 0;
for (var num = minimo; num<=maximo; num++) {
    if(AntepenultimaCifra(num) * UltimaCifra(num) * PenultimaCifra(num)
== AntepenultimaCifra(num) + UltimaCifra(num) + PenultimaCifra(num)){
        cuenta = cuenta + 1
    }
}
Imprime(minimo, maximo, "Cantidad de números que la multiplicación de las
tres últimas cifras es igual a la suma de esas tres últimas cifras",
cuenta);

//Imprime el resultado en el navegador
function Imprime(minimo, maximo, texto, resultado){
    document.write("Entre " + minimo + " y " + maximo + ". " + texto + ":"
+ resultado + "<br>");
}
</script></head></html>

```



Entre 1000 y 9999. Cantidad de números que la multiplicación de las tres últimas cifras es igual a la suma de esas tres últimas cifras: 63

Ilustración 58: Entre 1000 y 9999. Cantidad de números que la multiplicación de las tres últimas cifras es igual a la suma de esas tres últimas cifras: 63

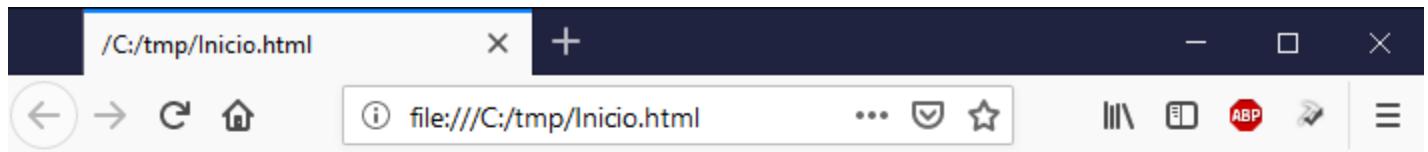
```

<!DOCTYPE HTML><html><head>
<script src="Libreria.js"></script>
<script>
var minimo = 8000;
var maximo = 9000;

//Cantidad de números que cada una de sus tres últimas cifras es menor de
5
var cuenta = 0;
for (var num = minimo; num<=maximo; num++) {
    if (AntepenultimaCifra(num) < 5 && PenultimaCifra(num) < 5 &&
UltimaCifra(num) < 5){
        document.write(num + ", ");
        cuenta = cuenta + 1
    }
}
document.write("<br>");
Imprime(minimo, maximo, "Cantidad de números que cada una de sus tres
últimas cifras es menor de 5", cuenta);

//Imprime el resultado en el navegador
function Imprime(minimo, maximo, texto, resultado){
    document.write("Entre " + minimo + " y " + maximo + ". " + texto + ":" +
" " + resultado + "<br>");
}
</script></head></html>

```



8000, 8001, 8002, 8003, 8004, 8010, 8011, 8012, 8013, 8014, 8020, 8021, 8022, 8023, 8024, 8030, 8031, 8032, 8033, 8034, 8040, 8041, 8042, 8043, 8044, 8100, 8101, 8102, 8103, 8104, 8110, 8111, 8112, 8113, 8114, 8120, 8121, 8122, 8123, 8124, 8130, 8131, 8132, 8133, 8134, 8140, 8141, 8142, 8143, 8144, 8200, 8201, 8202, 8203, 8204, 8210, 8211, 8212, 8213, 8214, 8220, 8221, 8222, 8223, 8224, 8230, 8231, 8232, 8233, 8234, 8240, 8241, 8242, 8243, 8244, 8300, 8301, 8302, 8303, 8304, 8310, 8311, 8312, 8313, 8314, 8320, 8321, 8322, 8323, 8324, 8330, 8331, 8332, 8333, 8334, 8340, 8341, 8342, 8343, 8344, 8400, 8401, 8402, 8403, 8404, 8410, 8411, 8412, 8413, 8414, 8420, 8421, 8422, 8423, 8424, 8430, 8431, 8432, 8433, 8434, 8440, 8441, 8442, 8443, 8444, 9000,

Entre 8000 y 9000. Cantidad de números que cada una de sus tres últimas cifras es menor de 5: 126

Ilustración 59: Entre 8000 y 9000. Cantidad de números que cada una de sus tres últimas cifras es menor de 5: 126

```

<!DOCTYPE HTML><html><head>
<script src="Libreria.js"></script>
<script>
var minimo = 8500;
var maximo = 8700;

//Cantidad de números que al juntar la antepenúltima cifra y la última
cifra se obtenga un primo
var contar = 0;
for (var num = minimo; num<=maximo; num++) {
    nuevo = AntepenultimaCifra(num) * 10 + UltimaCifra(num)
    if (EsPrimo(nuevo)) {
        document.write(num + ", ");
        contar = contar + 1
    }
}
document.write("<br>");
Imprime(minimo, maximo, "Cantidad de números que al juntar la
antepenúltima cifra y la última cifra se obtenga un primo", contar);

//Imprime el resultado en el navegador
function Imprime(minimo, maximo, texto, resultado) {
    document.write("Entre " + minimo + " y " + maximo + ". " + texto + ":" +
" " + resultado + "<br>");
}
</script></head></html>

```

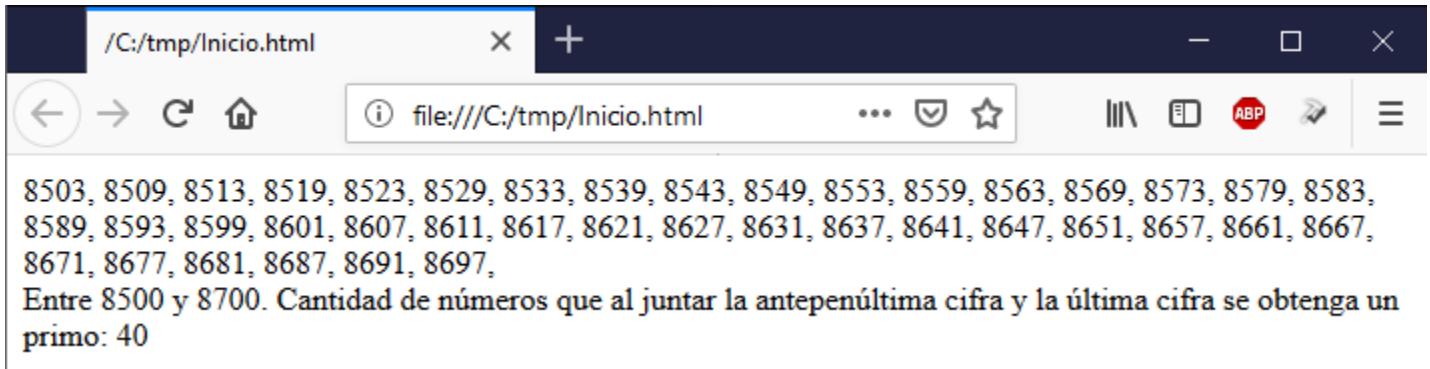


Ilustración 60: Entre 8500 y 8700. Cantidad de números que al juntar la antepenúltima cifra y la última cifra se obtenga un primo: 40

```

<!DOCTYPE HTML><html><head>
<script src="Libreria.js"></script>
<script>
var minimo = 4000;
var maximo = 7000;

//Cantidad de números primos que no tengan el número 3 en sus cifras
var contar = 0;
for (var num = minimo; num<=maximo; num++) {
    if (cifrashalladas(num, 3) == 0 && EsPrimo(num)) {
        document.write(num + ", ");
        contar = contar + 1
    }
}
document.write("<br>");
Imprime(minimo, maximo, "Cantidad de números primos que no tengan el
número 3 en sus cifras", contar);

//Imprime el resultado en el navegador
function Imprime(minimo, maximo, texto, resultado){
    document.write("Entre " + minimo + " y " + maximo + ". " + texto + ": "
    + resultado + "<br>");
}
</script></head></html>

```

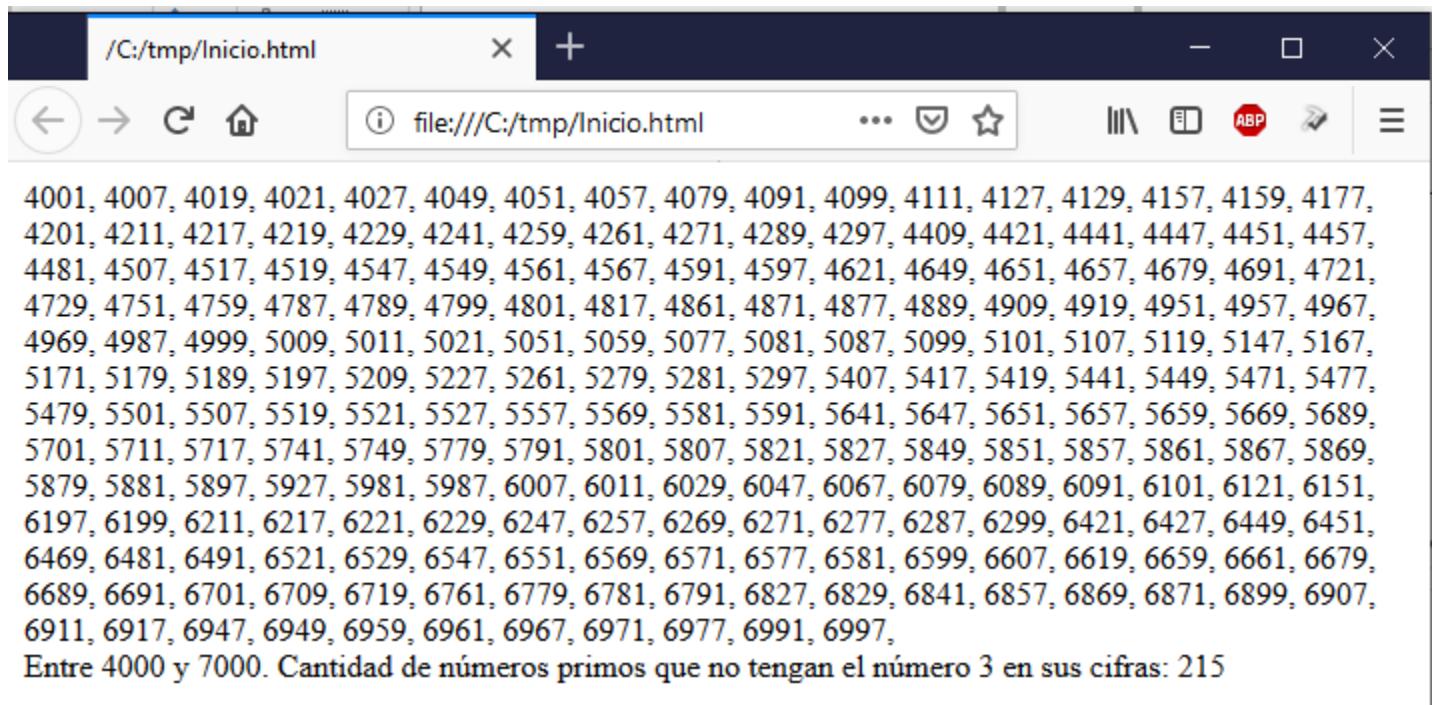


Ilustración 61: Entre 4000 y 7000. Cantidad de números primos que no tengan el número 3 en sus cifras: 215

```

<!DOCTYPE HTML><html><head>
<script src="Libreria.js"></script>
<script>
var minimo = 1000;
var maximo = 9999;

//Cantidad de números palíndromos que tengan mínimo dos veces el 7 en sus
cifras
var contar = 0;
for (var num = minimo; num<=maximo; num++) {
    if (cifrashalladas(num, 7) >= 2 && EsPalindromo(num)) {
        document.write(num + ", ");
        contar = contar + 1
    }
}
document.write("<br>");
Imprime(minimo, maximo, "Cantidad de números palíndromos que tengan mínimo
dos veces el 7 en sus cifras", contar);

//Imprime el resultado en el navegador
function Imprime(minimo, maximo, texto, resultado){
    document.write("Entre " + minimo + " y " + maximo + ". " + texto + ": "
    " + resultado + "<br>");
}
</script></head></html>

```



Ilustración 62: Entre 1000 y 9999. Cantidad de números palíndromos que tengan mínimo dos veces el 7 en sus cifras: 18

```

<!DOCTYPE HTML><html><head>
<script src="Libreria.js"></script>
<script>
var minimo = 5000;
var maximo = 9000;

//Cantidad de números que tengan solo cifras pares y al menos un número 4
var contar = 0;
for (var num = minimo; num<=maximo; num++) {
    if (cifrashalladas(num, 4) >= 1 && TodasCifrasPares(num)) {
        document.write(num + ", ");
        contar = contar + 1
    }
}
document.write("<br>");
Imprime(minimo, maximo, "Cantidad de números que tengan solo cifras pares
y al menos un número 4", contar);

//Imprime el resultado en el navegador
function Imprime(minimo, maximo, texto, resultado){
    document.write("Entre " + minimo + " y " + maximo + ". " + texto + ": "
    + resultado + "<br>");
}
</script></head></html>

```

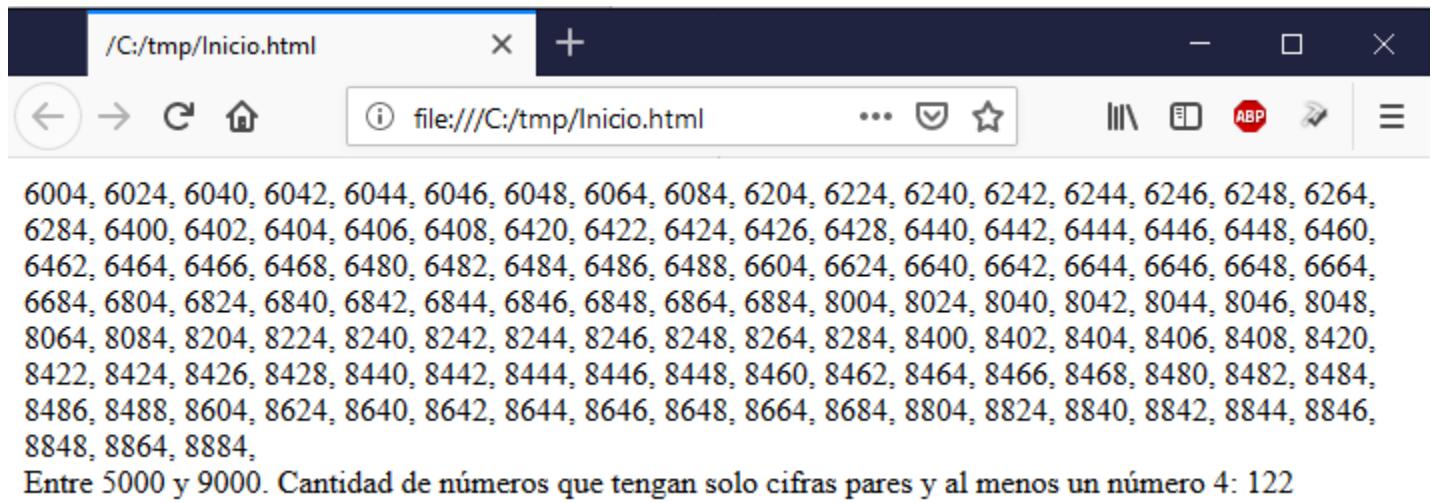


Ilustración 63: Entre 5000 y 9000. Cantidad de números que tengan solo cifras pares y al menos un número 4: 122

```

<!DOCTYPE HTML><html><head>
<script src="Libreria.js"></script>
<script>
var minimo = 7700;
var maximo = 8000;

//Cantidad de números que la suma de sus cifras es impar y al menos tenga
una vez el número 7
var contar = 0;
for (var num = minimo; num<=maximo; num++) {
    if (EsImpar(SumaCifras(num)) && cifrashalladas(num, 7) >= 1) {
        document.write(num + ", ");
        contar = contar + 1
    }
}
document.write("<br>");
Imprime(minimo, maximo, "Cantidad de números que la suma de sus cifras es
impar y al menos tenga una vez el número 7", contar);

//Imprime el resultado en el navegador
function Imprime(minimo, maximo, texto, resultado){
    document.write("Entre " + minimo + " y " + maximo + ". " + texto + ": "
    " + resultado + "<br>");
}
</script></head></html>

```



Ilustración 64: Entre 7700 y 8000. Cantidad de números que la suma de sus cifras es impar y al menos tenga una vez el número 7: 150

```

<!DOCTYPE HTML><html><head>
<script src="Libreria.js"></script>
<script>
var minimo = 1000;
var maximo = 9999;

//Cantidad de números primos que sus cifras están entre 3 y 7
var contar = 0;
for (var num = minimo; num<=maximo; num++) {
    if (EsPrimo(num) && LaCifraMasBaja(num) >= 3 && LaCifraMasAlta(num) <=
7) {
        document.write(num + ", ");
        contar = contar + 1
    }
}
document.write("<br>");
Imprime(minimo, maximo, "Cantidad de números primos que sus cifras están
entre 3 y 7", contar);

//Imprime el resultado en el navegador
function Imprime(minimo, maximo, texto, resultado){
    document.write("Entre " + minimo + " y " + maximo + ". " + texto + ": "
+ resultado + "<br>");
}
</script></head></html>

```

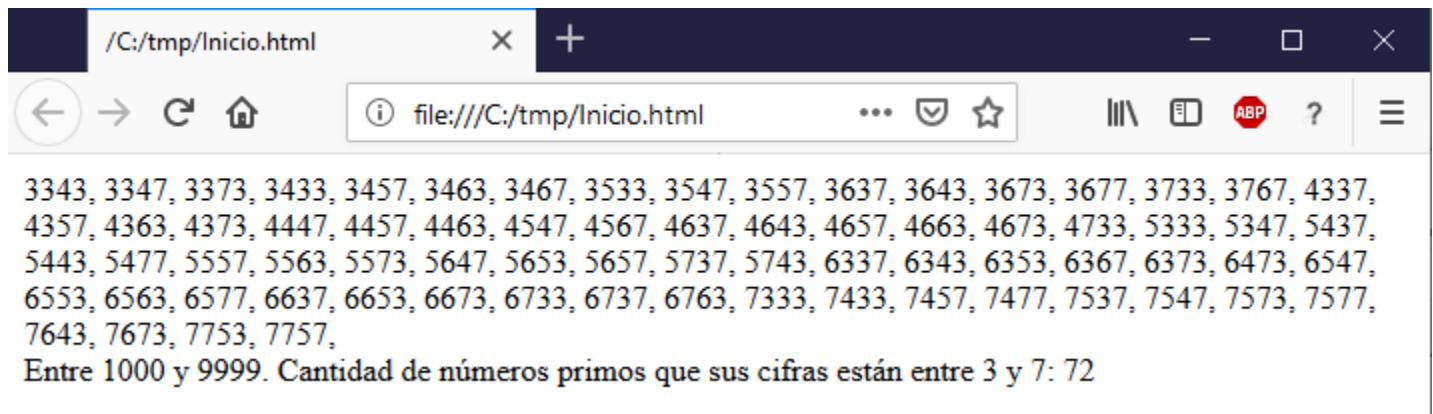


Ilustración 65: Entre 1000 y 9999. Cantidad de números primos que sus cifras están entre 3 y 7: 72

```

<!DOCTYPE HTML><html><head>
<script src="Libreria.js"></script>
<script>
var minimo = 10;
var maximo = 999999;

//Cantidad de números primos y a su vez palíndromos
var contar = 0;
for (var num = minimo; num<=maximo; num++) {
    if (EsPrimo(num) && EsPalindromo(num)) {
        document.write(num + ", ");
        contar = contar + 1
    }
}
document.write("<br>");
Imprime(minimo, maximo, "Cantidad de números primos y a su vez
palíndromos", contar);

//Imprime el resultado en el navegador
function Imprime(minimo, maximo, texto, resultado){
    document.write("Entre " + minimo + " y " + maximo + ". " + texto + ": "
    + resultado + "<br>");
}
</script></head></html>

```

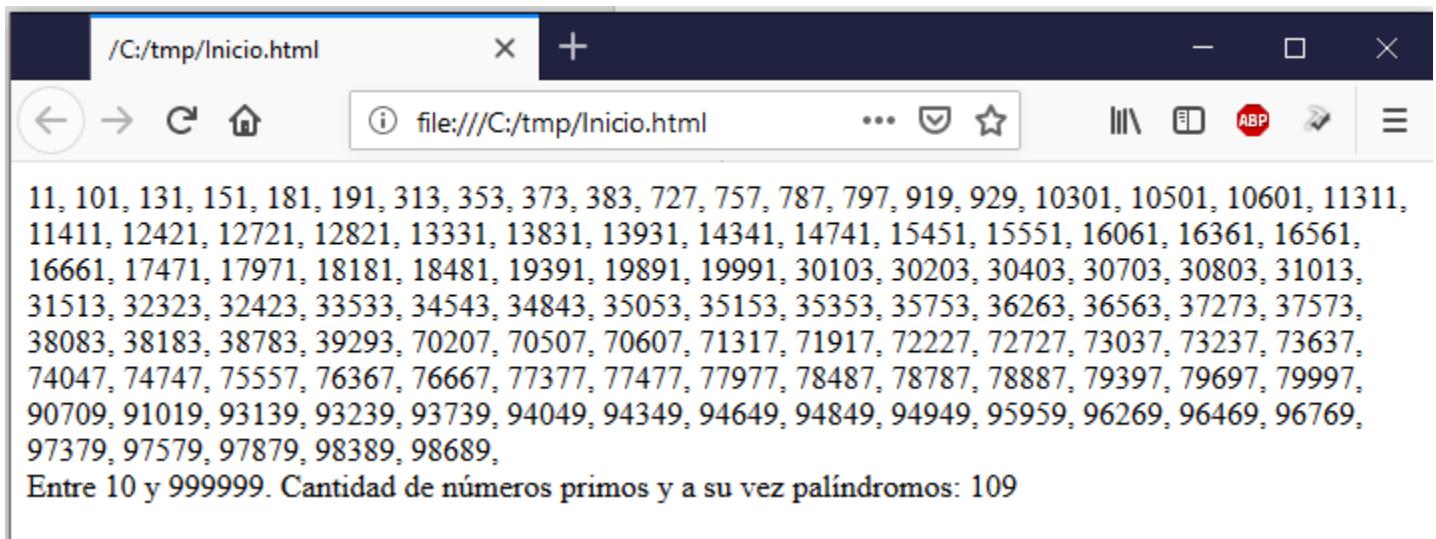


Ilustración 66: Entre 10 y 999999. Cantidad de números primos y a su vez palíndromos: 109

## Método de Bisección

Ver: [https://es.wikipedia.org/wiki/M%C3%A9todo\\_de\\_bisecci%C3%B3n](https://es.wikipedia.org/wiki/M%C3%A9todo_de_bisecci%C3%B3n)

Directorio 024

```
<!DOCTYPE HTML><html><head><script>
//Algoritmo de bisección
var X inicial = 1;
var X final = 2;
var Ciclos = 100; //Número de veces que hará la operación de aproximación

//Verifica que entre X inicial y X final exista un corte
if (ecuacion(X inicial) * ecuacion(X final) > 0){
    document.write("No hay punto de corte en los puntos dados");
} else{ //Si hay intersección, entonces llama a la función de Bisección
    var X resultado = Bisección(X inicial, X final, Ciclos);
    document.write("Punto de corte en: " + X resultado);
}

//Retorna el punto X entre X inicial y X final que más se aproxime al punto
//de corte
function Bisección(X inicial, X final, Ciclos){
    var X medio = 0;
    if (ecuacion(X inicial)==0) X medio = X inicial;
    else if (ecuacion(X final)==0) X medio = X final;
    else
        for(var cont=1; cont<=Ciclos; cont++){
            X medio = (X inicial + X final) / 2;
            if (ecuacion(X medio)==0) break;
            else if (ecuacion(X inicial)*ecuacion(X medio) < 0) X final =
X medio;
            else X inicial = X medio;
        }
    return X medio;
}

function ecuacion(x){ //En esta función se pone la ecuación
    return 4*x*x*x-3*x*x-5*x+2;
}
</script></head></html>
```

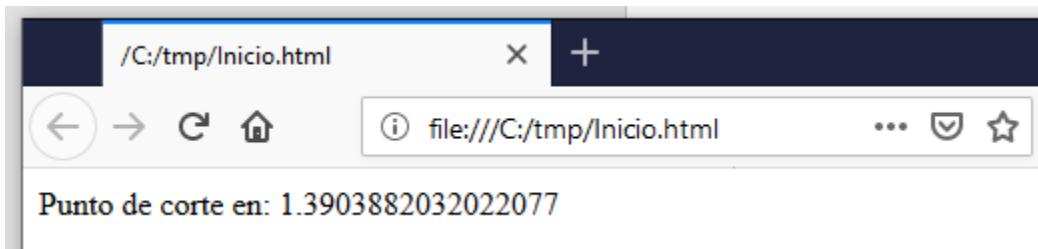


Ilustración 67: Método de bisección

# Ámbito (scope) de las variables y funciones

Directorio 025

```
<!DOCTYPE HTML><html><head><script>
//Ámbito (scope) de una variable
var valor = 17;

//Imprime 17
document.write("<br>Valor antes de la función es: " + valor);

prueba(); //Llama a la función

//Imprime 17
document.write("<br>Valor después de la función es: " + valor);

function prueba() {
    //Imprime 17
    document.write("<br>Valor dentro de la función es: " + valor);
}
</script></head></html>
```

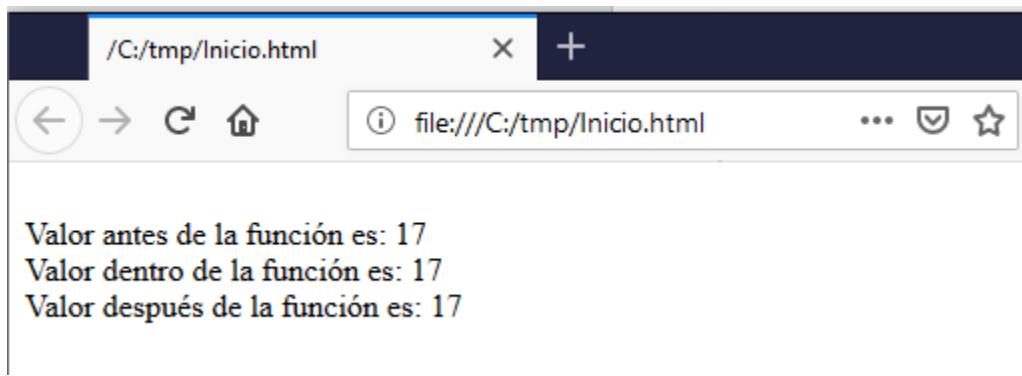


Ilustración 68: Ámbito (scope) de una variable

```
<!DOCTYPE HTML><html><head><script>
var valor = 17;
//Imprime 17
document.write("<br>Valor antes de la función es: " + valor);

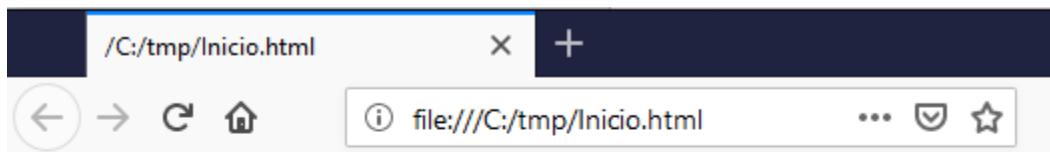
prueba(); //Llama a la función

//Imprime 8906
document.write("<br>Valor después de la función es: " + valor);

function prueba() {
    //Imprime 17
    document.write("<br>Valor dentro de la función es: " + valor);

    valor = 8906;

    //Imprime 8906
    document.write("<br>Valor después de cambiar es: " + valor);
}
</script></head></html>
```



Valor antes de la función es: 17  
Valor dentro de la función es: 17  
Valor después de cambiar es: 8906  
Valor después de la función es: 8906

Ilustración 69: Ámbito (scope) de una variable

```
<!DOCTYPE HTML><html><head><script>
var valor = 17;

//Imprime 17
document.write("<br>Valor antes de la función es: " + valor);

prueba(); //Llama a la función

//Imprime 17
document.write("<br>Valor después de la función es: " + valor);

function prueba() {
    //Imprime undefined
    document.write("<br>Valor dentro de la función es: " + valor);

    var valor = 8906;

    //Imprime 8906
    document.write("<br>Valor variable interna: " + valor);
}

</script></head></html>
```



Ilustración 70: Ámbito (scope) de una variable

```
<!DOCTYPE HTML><html><head><script>
//Ámbito (scope) de una variable
var valor = 17;

//Imprime 17
document.write("<br>Valor antes de la función es: " + valor);

prueba(valor); //Llama a la función enviando por parámetro el valor

//Imprime 17
document.write("<br>Valor después de la función es: " + valor);

function prueba(valor){
    //Imprime 17
    document.write("<br>Valor dentro de la función es: " + valor);

    var valor = 8906;

    //Imprime 8906
    document.write("<br>Valor variable interna: " + valor);
}

</script></head></html>
```

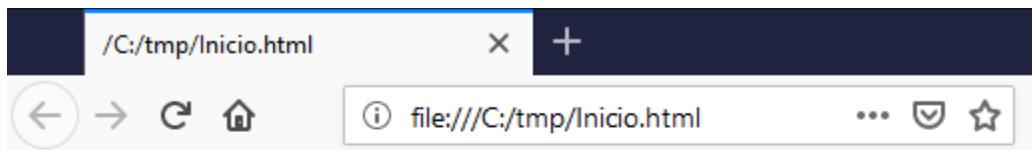


Ilustración 71: Ámbito (scope) de una variable

## Funciones recursivas

Funciones que se llaman a sí mismas y deben tener un criterio de salida.

Directorio 026

```
<!DOCTYPE HTML><html><head><script>
/* Función recursiva
   Factorial(5) = 5 * Factorial(4)
     Es decir Factorial(N) = N * Factorial(N-1);
     y Factorial(0) = 1
*/
function factorial(numero) {
    if (numero > 1)
        return numero * factorial(numero-1)
    else
        return 1;
}

document.write("5! = " + factorial(5));
</script></head></html>
```

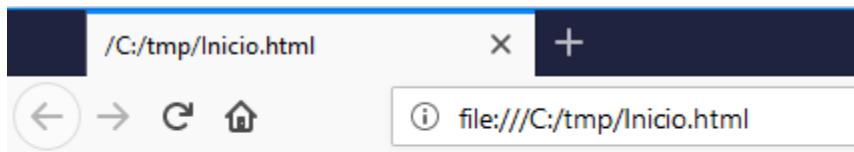


Ilustración 72: Cálculo del factorial con función recursiva

Directorio 026

```
<!DOCTYPE HTML><html><head><script>
// Función recursiva. Máximo común divisor con algoritmo de Euclides
function maximocomundivisor(numA, numB) {
    if(numA < numB) return maximocomundivisor(numB, numA);
    if(numB == 0) return numA;
    return maximocomundivisor(numB, numA % numB);
}

var numero = maximocomundivisor(1000, 750);
document.write("Máximo común divisor de 1000 y 750 es " + numero);
</script></head></html>
```



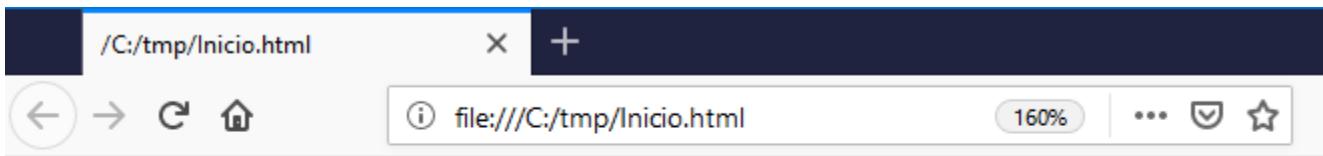
Máximo común divisor de 1000 y 750 es 250

Ilustración 73: Cálculo del máximo común divisor con función recursiva

Directorio 026

```
<!DOCTYPE HTML><html><head><script>
// Función recursiva. Suma las cifras de un número
function sumacifras(num) {
    if (num < 10) return num;
    return num%10 + sumacifras(Math.floor(num/10));
}

document.write("cifras suman: " + sumacifras(701));
</script></head></html>
```



cifras suman: 8

Ilustración 74: Suma de cifras con función recursiva

Directorio 026

```
<!DOCTYPE HTML><html><head><script>
//Convertir a binario
ConvierteBinario(70);

function ConvierteBinario(numero){
    if (numero!=0){
        ConvierteBinario(Math.floor(numero/2));
        document.write(numero%2);
    }
}
</script></head></html>
```

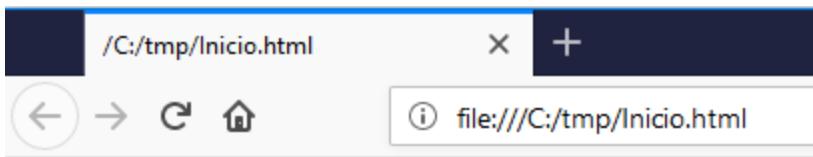


Ilustración 75: Conversión a número binario con función recursiva

Directorio 026

```
<!DOCTYPE HTML><html><head><script>
//Potencia de un número
var valor = potencia(2, 7); //Retorna 2^7
document.write(valor);

function potencia(numero, elevado){
    if (elevado==1) return numero;
    return numero * potencia(numero, elevado - 1);
}
</script></head></html>
```

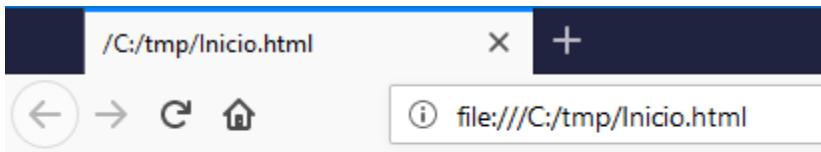
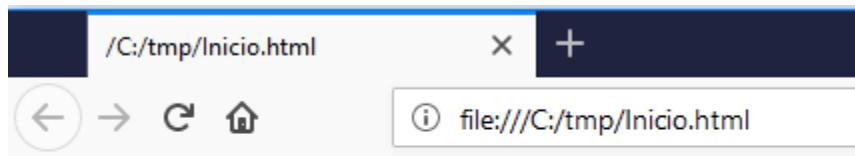


Ilustración 76: Cálculo de potencia con función recursiva

## Uso de eval como evaluador de expresiones algebraicas

Directorio 027

```
<!DOCTYPE HTML><html><head><script>
//Uso de eval como evaluador de expresiones algebraicas
var valor = eval("3+2*5");
document.write(valor);
</script></head></html>
```

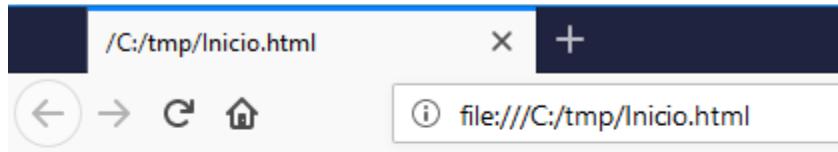


13

Ilustración 77: Eval como evaluador de expresiones

Directorio 027

```
<!DOCTYPE HTML><html><head><script>
//Uso de eval como evaluador de expresiones algebraicas
var valor = eval("Math.sin(15)-Math.cos(21)");
document.write(valor);
</script></head></html>
```



1.198017100381385

Ilustración 78: Eval como evaluador de expresiones

```
<!DOCTYPE HTML><html><head><script>
//Uso de eval como evaluador de expresiones algebraicas
var a=3;
var b=5;
var c=2;
var d=7;
var valor = eval("a+b-c*d");
document.write(valor);
</script></head></html>
```

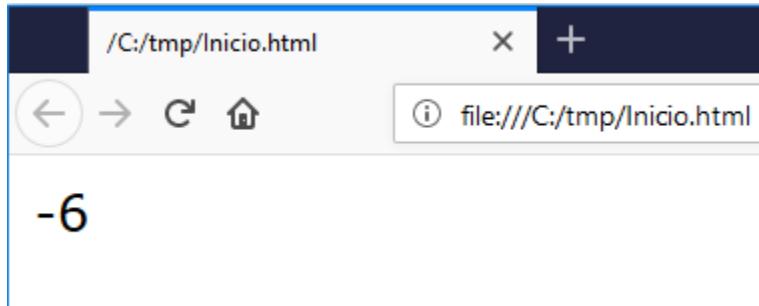


Ilustración 79: Eval como evaluador de expresiones

## Números aleatorios

Directorio 028

```
<!DOCTYPE HTML><html><head><script>
//Números aleatorios
var numero = Math.random(); //Un número entre 0 y 1. Nunca dará 1.
document.write(numero);
</script></head></html>
```

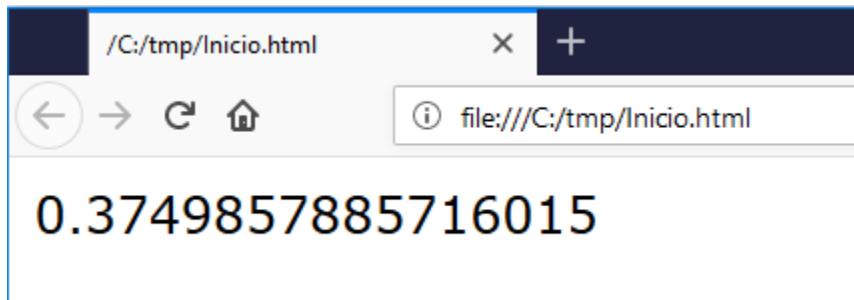


Ilustración 80: Número aleatorio

Directorio 028

```
<!DOCTYPE HTML><html><head><script>
//Números aleatorios. Ciclo.
for (var num = 1; num <= 100; num++)
    document.write(Math.random() + ", ");
</script></head></html>
```

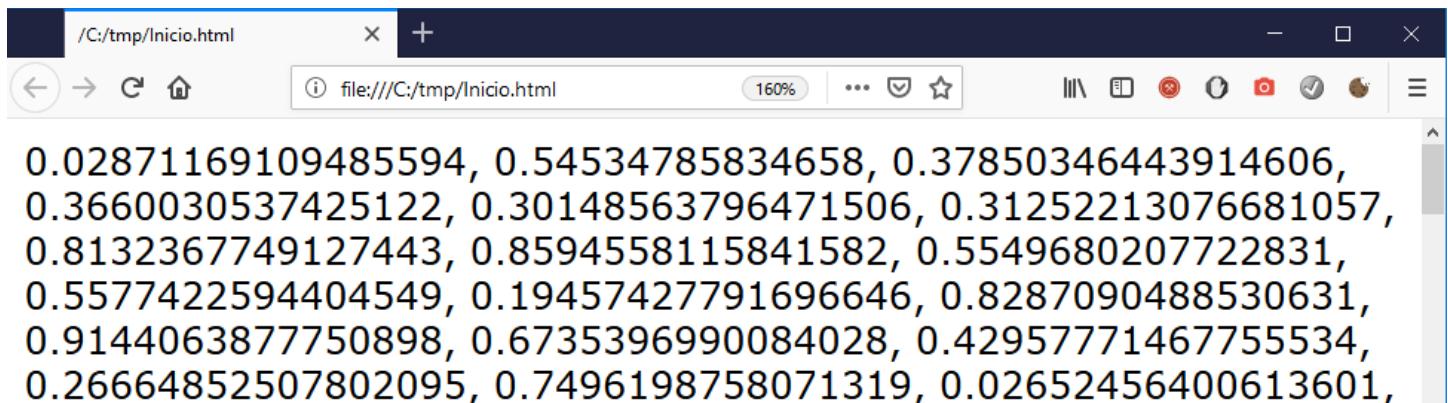


Ilustración 81: Listado de números aleatorios

## Distribución Normal

Directorio 028

```
<!DOCTYPE HTML><html><head><script>
/* Variable aleatoria
Distribución Normal. Generar una variable aleatoria con media M y
desviación D
variable = M + D * c
donde c = cos(2*PI*r2)*raizcuadrada(-2*LogarimoNatural(r1))
r1 y r2 son números aleatorios */
var M = 100;
var D = 7;
for (var cont=1; cont<=100; cont++) {
    var r1 = Math.random();
    var r2 = Math.random();
    var c = Math.cos(2*Math.PI*r2)*Math.sqrt(-2*Math.log(r1));
    var variable = M + D * c;
    document.write(variable + ", ");
}
</script></head></html>
```

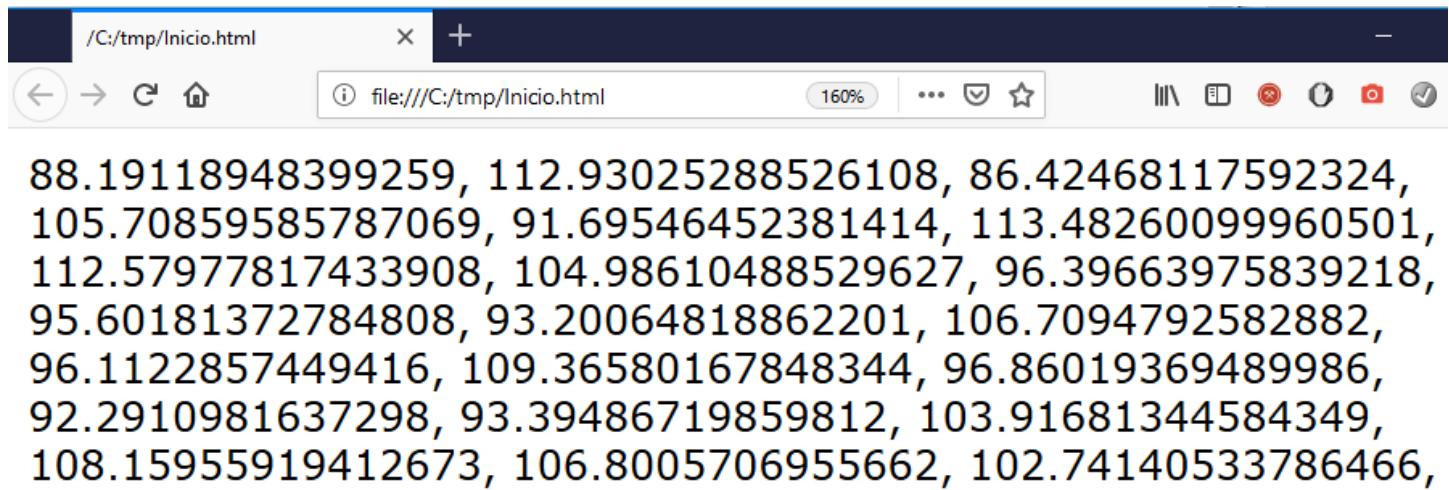


Ilustración 82: Distribución normal

## Distribución Triangular

Directorio 028

```
<!DOCTYPE HTML><html><head><meta charset="UTF-8"><script>
/* Distribución Triangular. Generar una variable aleatoria con valor
mínimo A,
    valor más probable B y valor máximo C
    Si r < (B-A) / (C-A)
        Variable = A + raizcuadrada(r * (C-A) * (B-A))
    de lo contrario
        Variable = C - raizcuadrada((1-r) * (C-A) * (C-B))
    */
var A = 150; var B = 190; var C = 230; var variable = 0;
for (var cont=1; cont<=100; cont++) {
    var r = Math.random();
    if (r < (B-A)/(C-A))
        variable = A + Math.sqrt(r*(C-A)*(B-A));
    else
        variable = C - Math.sqrt((1-r)*(C-A)*(C-B));
    document.write(variable + ", ");
}
</script></head></html>
```

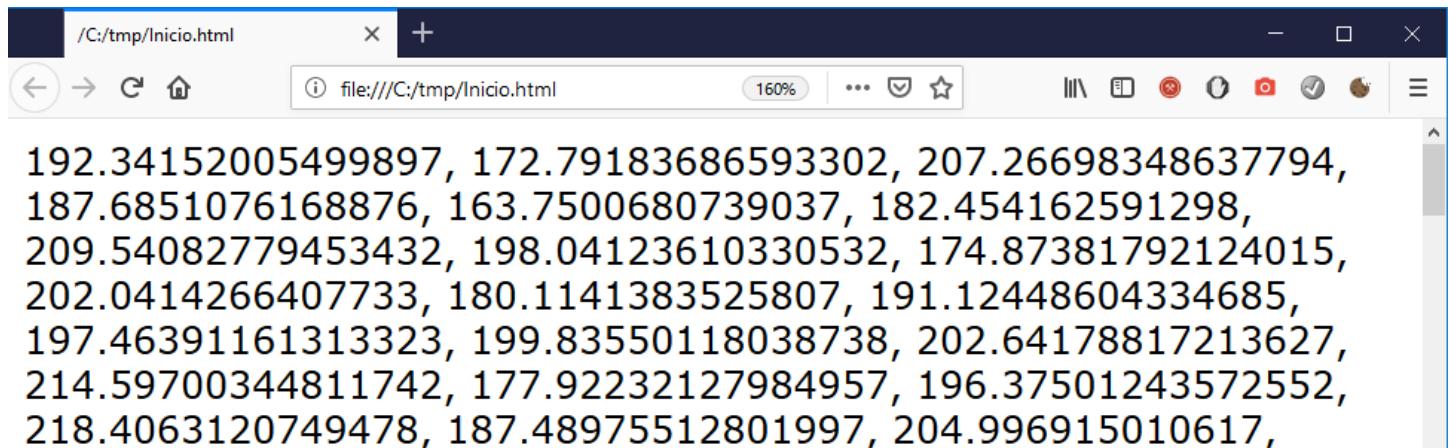


Ilustración 83: Distribución triangular

## Distribución Uniforme

Directorio 028

```
<!DOCTYPE HTML><html><head><script>
/* Variable aleatoria
Distribución uniforme. Generar un número entero entre A y B
variable = r * (B - A) + A */
var A = 10;
var B = 50;
for (var cont=1; cont<=100; cont++) {
    var r = Math.random();
    var variable = r * (B - A) + A;
    document.write(variable + ", ");
}
</script></head></html>
```

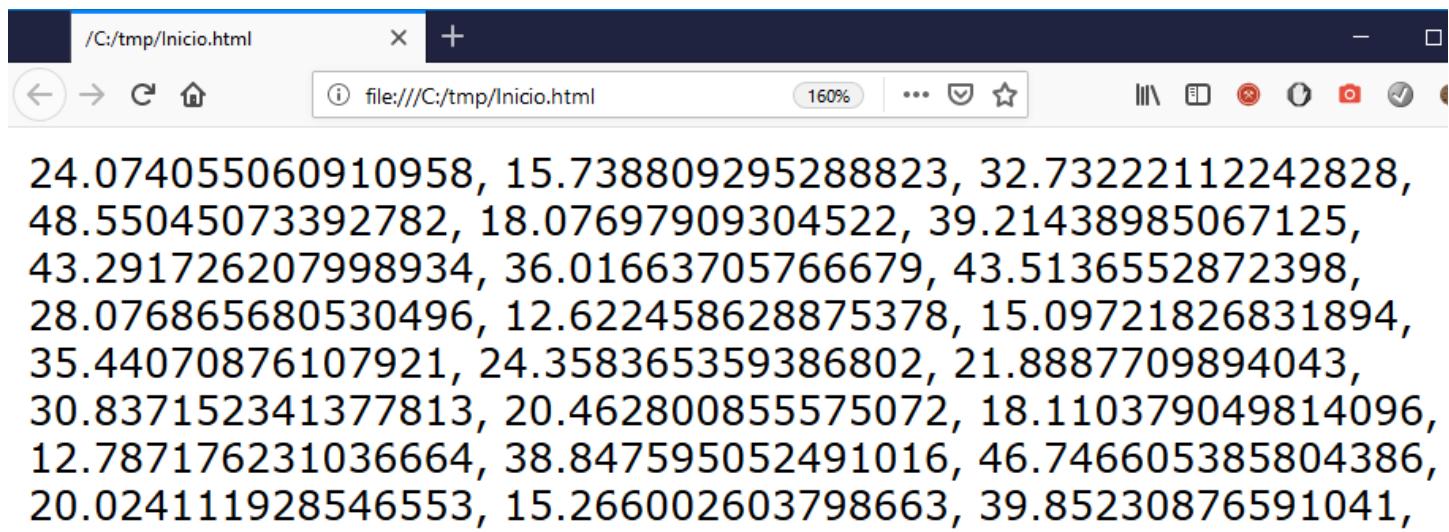


Ilustración 84: Distribución uniforme

## Manejo de errores con números

### isNaN

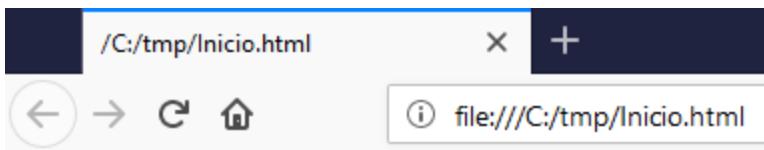
isNaN significa “is Not a Number”, si lo que entra por parámetro no es un número retorna true. Más información: <https://es.wikipedia.org/wiki/NaN>

Directorio 029

```
<!DOCTYPE HTML><html><head><script>
//Chequea si el usuario digitó un número correctamente
var numero = parseInt(prompt("Digite número"));
if (isNaN(numero))
    document.write("No escribió un número");
else
    document.write("Número escrito es: " + numero);
</script></head></html>
```

Directorio 029

```
<!DOCTYPE HTML><html><head><script>
//Uso de isNaN
document.write(isNaN(891)+"<br>"); //false
document.write(isNaN(-7.19)+"<br>"); //false
document.write(isNaN(79+21)+"<br>"); //false
document.write(isNaN(0)+"<br>"); //false
document.write(isNaN('888')+"<br>"); //false
document.write(isNaN('Probar')+"<br>"); //true
document.write(isNaN('2019/01/13')+"<br>"); //true
document.write(isNaN('')+ "<br>"); //false
document.write(isNaN(true)+"<br>"); //false
document.write(isNaN(undefined)+"<br>"); //true
document.write(isNaN('NaN')+"<br>"); //true
document.write(isNaN(NaN)+"<br>"); //true
document.write(isNaN(0 / 0)+"<br>"); //true
</script></head></html>
```



false  
false  
false  
false  
false  
true  
true  
false  
false  
true  
true  
true  
true

Ilustración 85: Uso de isNaN

Directorio 029

```
<!DOCTYPE HTML><html><head><script>
//Chequea si existe el NaN (Not a Number)
var numero = parseFloat(prompt("Digite número"));
var valor = Math.asin(numero);
if (isNaN(valor))
    document.write("Valor erróneo para arcoseno");
else
    document.write("Arcoseno es: " + valor);
</script></head></html>
```

## isFinite

Una división entre cero daría infinito, luego isFinite lo detecta.

Directorio 029

```
<!DOCTYPE HTML><html><head><script>
//Verificar si un número es legal o no
var dividendo = parseInt(prompt("Digite dividendo"));
var divisor = parseInt(prompt("Digite divisor"));
var resultado = dividendo / divisor;
if (isFinite(resultado)) //Verifica si el resultado es un número legal
    document.write("Valor correcto: " + resultado);
else
    document.write("Número no legal: " + resultado);
</script></head></html>
```

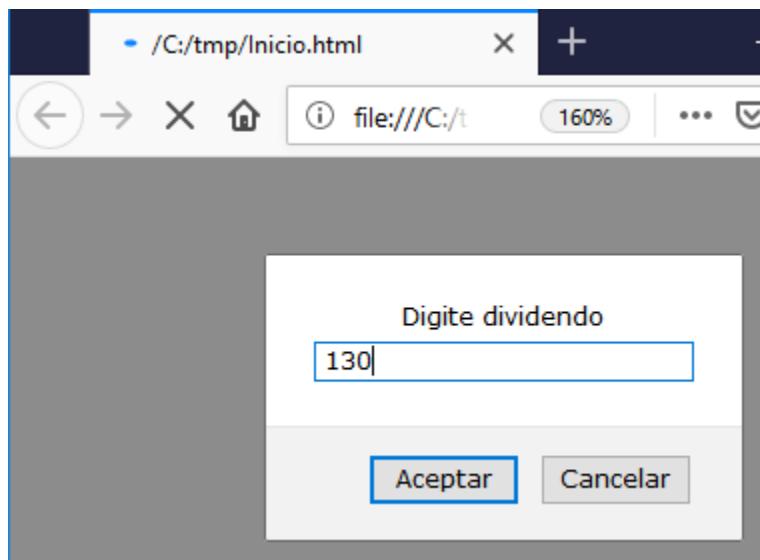


Ilustración 86: Ingresa dividendo

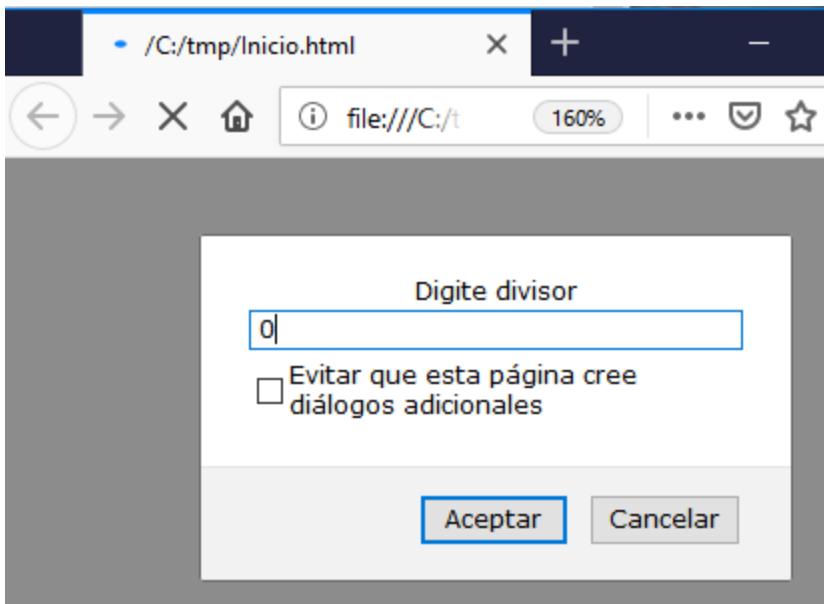


Ilustración 87: Ingresa divisor como cero para generar un error

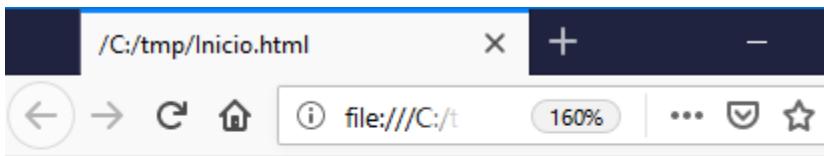


Ilustración 88: La división entre cero da "Infinity"

## Try...catch

Captura el error matemático

Directorio 029

```
<!DOCTYPE HTML><html><head><script>
//Captura el error
var expresion = prompt("Escriba expresión algebraica");
try {
    var valor = eval(expresion);
    document.write("Valor calculado es: " + valor);
}
catch(errordetectado){
    document.write("Mensaje de error: " + errordetectado.message);
}
</script></head></html>
```

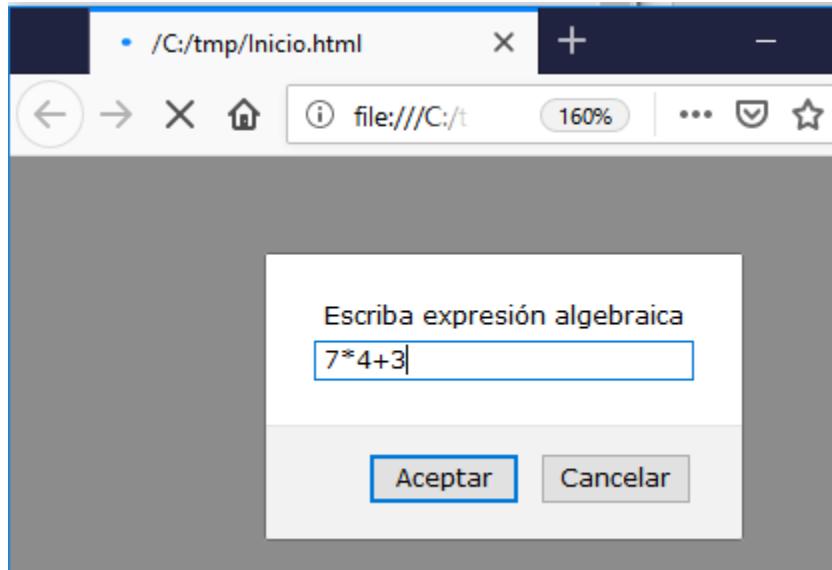


Ilustración 89: Ingresa expresión válida

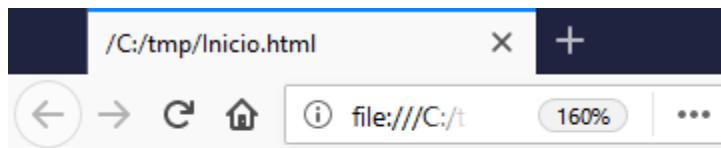


Ilustración 90: Hace el cálculo con expresión válida

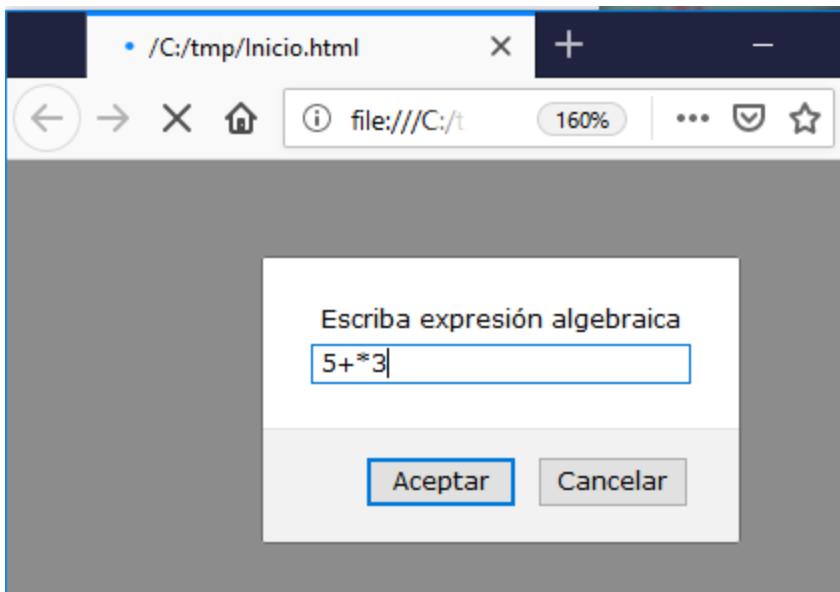


Ilustración 91: Ingresa expresión con sintaxis errónea

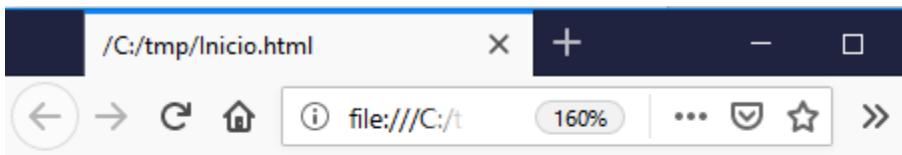


Ilustración 92: Mensaje de error por la sintaxis errónea

## Arreglos unidimensionales o vectores

JavaScript trae varias funciones para el trabajo con arreglos unidimensionales. En JavaScript los arreglos unidimensionales siempre inician en la posición cero.

Directorio 030

```
<!DOCTYPE HTML><html><head><script>
var lenguajes = new Array(); //Vectores o arreglos
lenguajes.push("C++"); //con "push" adiciona elementos al arreglo
lenguajes.push("C#");
lenguajes.push("Visual Basic .Net");
lenguajes.push("PHP");
lenguajes.push("JavaScript");
lenguajes.push("Object Pascal");
document.write(lenguajes); //Imprime el arreglo unidimensional
</script></head></html>
```

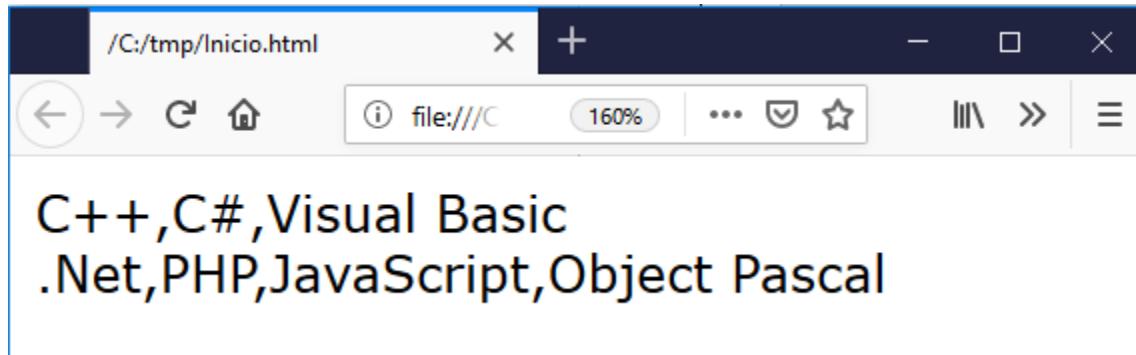
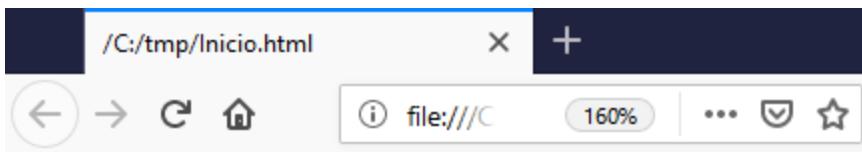


Ilustración 93: Arreglo unidimensional impreso

Directorio 030

```
<!DOCTYPE HTML><html><head><script>
var lenguajes = new Array(); //Vectores o arreglos
lenguajes.push("C++");
lenguajes.push("C#");
lenguajes.push("Visual Basic .Net");
lenguajes.push("PHP");
lenguajes.push("JavaScript");
lenguajes.push("Object Pascal");

//Recorrer un vector elemento por elemento
for (var cont=0; cont<lenguajes.length; cont++){
    document.write("<br>" + lenguajes[cont]);
}
</script></head></html>
```

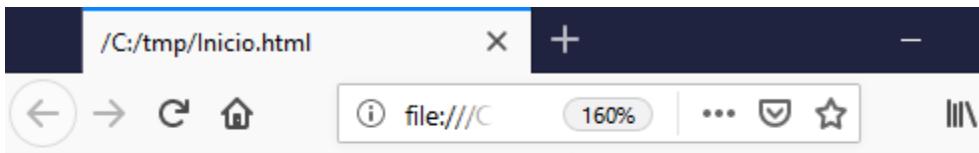


C++  
C#  
Visual Basic .Net  
PHP  
JavaScript  
Object Pascal

Ilustración 94: Muestra ítem a ítem el arreglo unidimensional

Directorio 030

```
<!DOCTYPE HTML><html><head><script>
var lenguajes = []; //Otra forma de declarar arreglos
lenguajes.push("C++"); //con "push" adiciona elementos al arreglo
lenguajes.push("C#");
lenguajes.push("Visual Basic .Net");
lenguajes.push("PHP");
lenguajes.push("JavaScript");
lenguajes.push("Object Pascal");
document.write(lenguajes); //Imprime el arreglo unidimensional
</script></head></html>
```



C++,C#,Visual Basic  
.Net,PHP,JavaScript,Object Pascal

Ilustración 95: Otra forma de declarar arreglo unidimensional

```
<!DOCTYPE HTML><html><head><script>
//Otra forma de declarar arreglos y mostrarlos
var pagar = [1120, 6040, 1570, 5205, 3147, 7361, 166, 1002, 1572, 6567];
document.write(pagar);
</script></head></html>
```

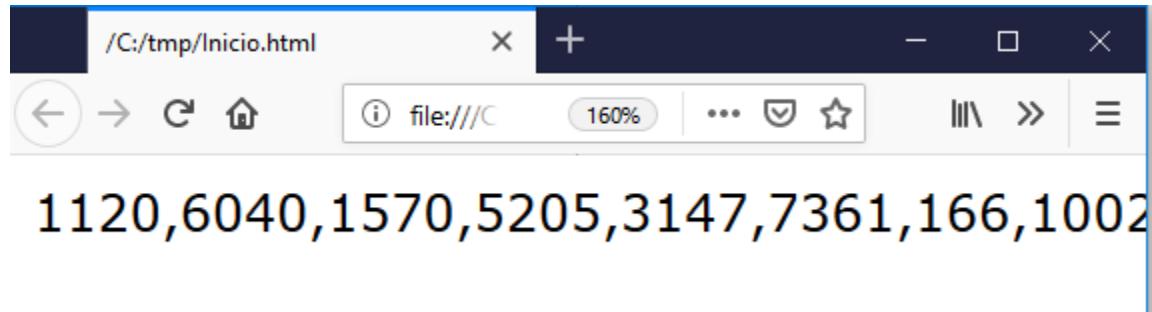


Ilustración 96: Otra forma de declarar arreglo unidimensional

## Borrar elementos con splice()

Directorio 030

```
<!DOCTYPE HTML><html><head><script>
var lenguajes = new Array(); //Vectores o arreglos
lenguajes.push("C++"); //con "push" adiciona elementos al arreglo
lenguajes.push("C#");
lenguajes.push("Visual Basic .Net");
lenguajes.push("PHP");
lenguajes.push("JavaScript");
lenguajes.push("Object Pascal");
document.write(lenguajes); //Imprime el arreglo unidimensional

document.write("<br>");

//Desde la posición 1 borre 3 ítems
lenguajes.splice(1, 3);
document.write(lenguajes);
</script></head></html>
```



Ilustración 97: Uso de splice() para borrar ítems en un arreglo

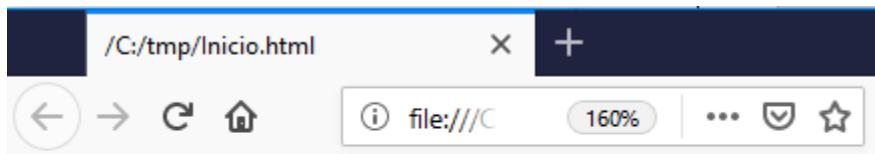
## Operaciones con arreglos

Directorio 030

```
<!DOCTYPE HTML><html><head><script>
var valores = new Array();
valores.push(15000);
valores.push(18000);
valores.push(34000);
valores.push(17000);
valores.push(8000);

//Operación con vectores: acumulado
var acumula = 0;
for (var cont=0; cont < valores.length; cont++) {
    acumula += valores[cont];
}

document.write("Acumulado es: " + acumula);
</script></head></html>
```



Acumulado es: 92000

Ilustración 98: Operación matemática con elementos del arreglo unidimensional

Directorio 030

```
<!DOCTYPE HTML><html><head><script>
//Operación con vectores: promedio
var valores = new Array();
valores.push(15000);
valores.push(18000);
valores.push(34000);
valores.push(17000);
valores.push(8000);

//Acumula los valores
var acumula = 0;
for (var cont=0; cont < valores.length; cont++) {
    acumula += valores[cont];
}

//Calcula el promedio
var promedio = acumula / valores.length;
document.write("Promedio es: " + promedio);
```

```
</script></head></html>
```

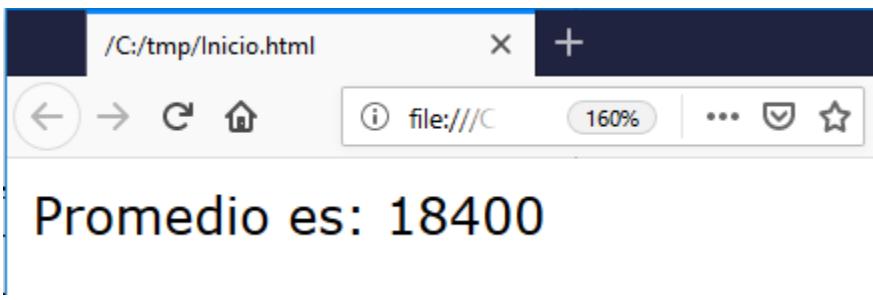


Ilustración 99: Operación matemática con elementos del arreglo unidimensional

Directorio 030

```
<!DOCTYPE HTML><html><head><script>
//Operación con vectores: Buscar el mayor valor
var valores = new Array();
valores.push(15000);
valores.push(18000);
valores.push(34000);
valores.push(17000);
valores.push(8000);

var mayor = valores[0]; //Inicia con el primer elemento como el mayor
for (var cont=0; cont < valores.length; cont++) {
    if (valores[cont] > mayor) mayor = valores[cont];
}

document.write("Mayor es: " + mayor);
</script></head></html>
```

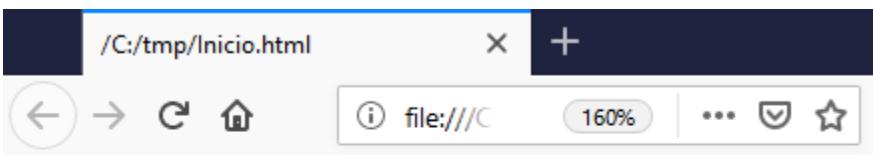


Ilustración 100: Operación matemática con elementos del arreglo unidimensional

## Ordenación de arreglo unidimensional

Directorio 030

```
<!DOCTYPE HTML><html><head><script>
//Ordena arreglos con valores alfanuméricos
var lenguajes = new Array();
lenguajes.push("Visual Basic .Net");
lenguajes.push("PHP");
lenguajes.push("JavaScript");
lenguajes.push("Java");
lenguajes.push("Object Pascal");
lenguajes.push("C#");

document.write("<b>Arreglo original:</b> " + lenguajes);

lenguajes.sort(); //Ordena el arreglo en orden alfabético

document.write("<br><b>Arreglo ordenado:</b> " + lenguajes);
</script></head></html>
```

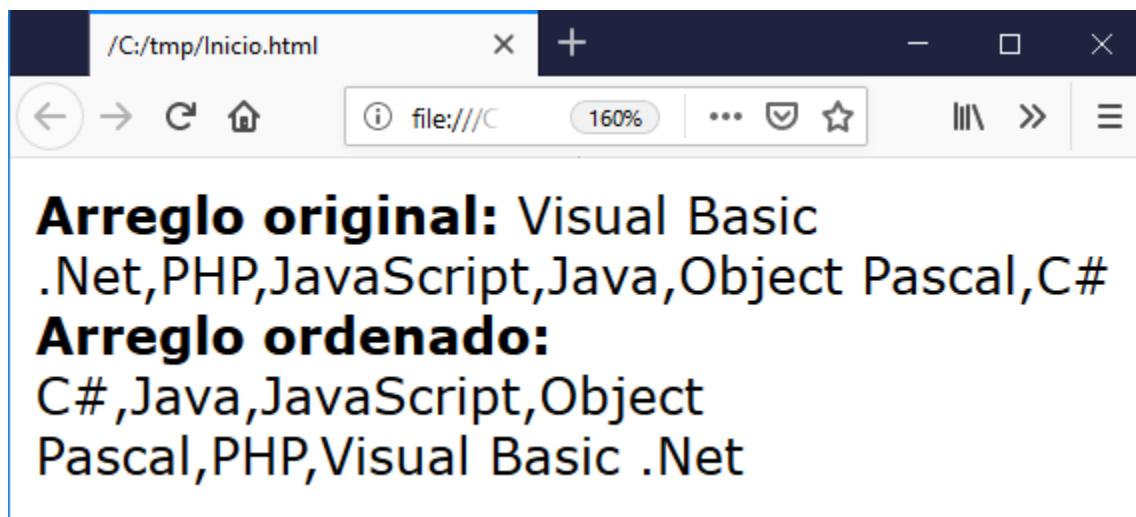


Ilustración 101: Ordenación de arreglo unidimensional

```
<!DOCTYPE HTML><html><head><script>
//Ordenar de menor a mayor el arreglo con valores numéricos
var valores = new Array();
valores.push(19000);
valores.push(18000);
valores.push(34000);
valores.push(27000);
valores.push(8000);
valores.push(9000);
document.write("Arreglo numérico original: " + valores);

valores.sort(function(a,b){return a - b}); //Ordena el arreglo
document.write("<br>Arreglo numérico ordenado: " + valores);
</script></head></html>
```

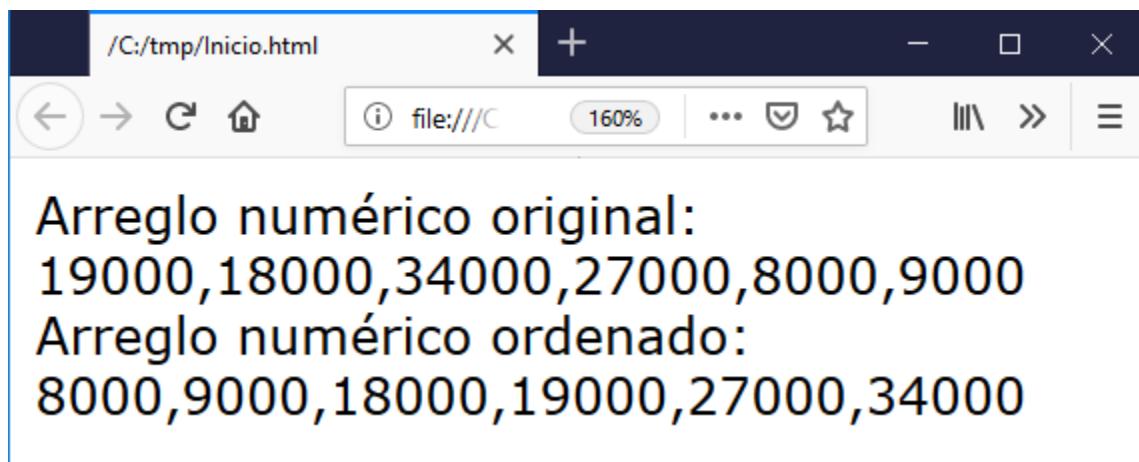


Ilustración 102: Ordenación de arreglo unidimensional

```
<!DOCTYPE HTML><html><head><script>
//Ordena en forma inversa un arreglo con valores alfanuméricicos
var lenguajes = new Array();
lenguajes.push("Visual Basic .Net");
lenguajes.push("PHP");
lenguajes.push("JavaScript");
lenguajes.push("Java");
lenguajes.push("Object Pascal");
lenguajes.push("C#");
document.write("<b>Arreglo original:</b> " + lenguajes);
lenguajes.sort(); //Ordena el arreglo en orden alfabético
document.write("<br><b>Arreglo ordenado:</b> " + lenguajes);
lenguajes.reverse(); //Invierte el arreglo
document.write("<br><b>Arreglo invertido:</b> " + lenguajes);
</script></head></html>
```

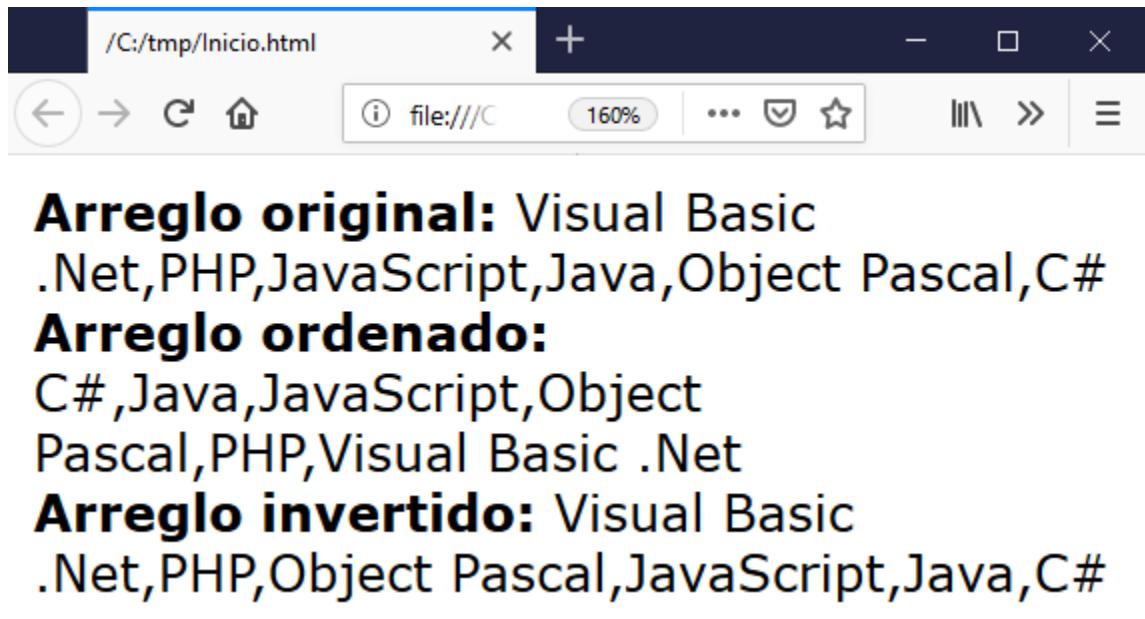


Ilustración 103: Ordenación de arreglo de menor a mayor, y de mayor a menor

## Funciones genéricas para arreglos

Directorio 030

```
<!DOCTYPE HTML><html><head><script>
//Dado un arreglo extraer los valores únicos
var numeros = [5,8,1,3,3,2,7,8,1,9,9,1,1,2,4,7,2,3,8,4,3,1,1,3,2,8,8,7];
var extrae = new Array();

for (var cont=0; cont<numeros.length; cont++)
    if ( BuscarEnArreglo( numeros[cont], extrae) == false )
        extrae.push( numeros[cont] );

document.write(extrae);

function BuscarEnArreglo(num, arreglo){ //Retorna true si encuentra num en
arreglo
    for (var cont=0; cont<arreglo.length; cont++)
        if (num==arreglo[cont])
            return true;
    return false;
}
</script></head></html>
```

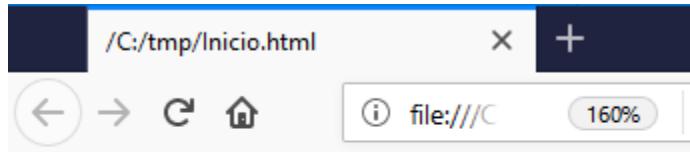


Ilustración 104: Dado un arreglo extraer los valores únicos

```
<!DOCTYPE HTML><html><head><script>
// ¿Función genérica para hallar menor y mayor valor?
var pagar = [1120, 6040, 1570, 5205, 3147, 7361, 166, 1002, 1572, 6567];

var menor = BuscaMenorValor(pagar);
var mayor = BuscaMayorValor(pagar);
document.write("Menor cuantía es: " + menor);
document.write("<br>Mayor cuantía es: " + mayor);

function BuscaMenorValor(arreglo){ //Función genérica
    var minimo = arreglo[0];
    for(var cont=1; cont < arreglo.length; cont++)
        if (arreglo[cont] < minimo) minimo = arreglo[cont];
    return minimo;
}

function BuscaMayorValor(arreglo){ //Función genérica
    var maximo = arreglo[0];
    for(var cont=1; cont < arreglo.length; cont++)
        if (arreglo[cont] > maximo) maximo = arreglo[cont];
    return maximo;
}
</script></head></html>
```

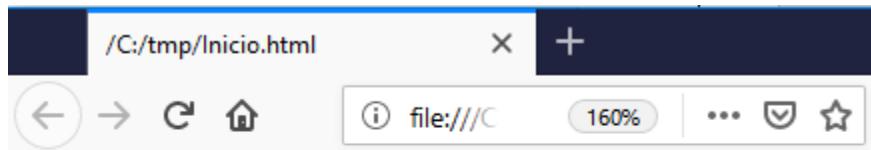


Ilustración 105: ¿Función genérica para hallar menor y mayor valor?

## Retornar arreglos desde funciones

Directorio 030

```
<!DOCTYPE HTML><html><head><script>
// Separe los elementos pares e impares de una lista, y retorne ambas
listas
var numeros = [10, 13, 20, 24, 23, 2, 23, 29, 5, 19, 8, 6, 16, 2, 6, 27];
var pares = retornaPares(numeros);
var impares = retornaImpares(numeros);
document.write("Pares: " + pares + "<br>");
document.write("Impares: " + impares + "<br>");

function retornaPares(valores){
    var par = new Array();
    for (var cont=0; cont<valores.length; cont++)
        if (valores[cont]%2==0)
            par.push(valores[cont]);
    return par;
}

function retornaImpares(valores){
    var impar = new Array();
    for (var cont=0; cont<valores.length; cont++)
        if (valores[cont]%2!=0)
            impar.push(valores[cont]);
    return impar;
}
</script></head></html>
```

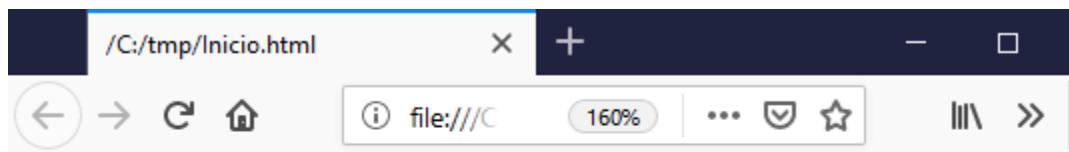


Ilustración 106: Separe los elementos pares e impares de una lista, y retorne ambas listas

# Implementación de algoritmos de ordenación

## Algoritmo de burbuja

Directorio 031

```
<!DOCTYPE HTML><html><head><script>
var letras = ['r', 'v', 'e', 'g', 't', 'u', 'd', 'f', 'f', 'y', 'v'];
document.write("Original: " + letras + "<br>");
OrdenarBurbuja(letras);
document.write("Ordenado: " + letras + "<br>");

//Ordenación por el método de burbuja
function OrdenarBurbuja(arreglo){
    for (var i=0; i<arreglo.length-1; i++)
        for (var j=0; j<arreglo.length-i-1; j++)
            if (arreglo[j]>arreglo[j+1]){
                var aux = arreglo[j];
                arreglo[j] = arreglo[j+1];
                arreglo[j+1] = aux;
            }
}
</script></head></html>
```

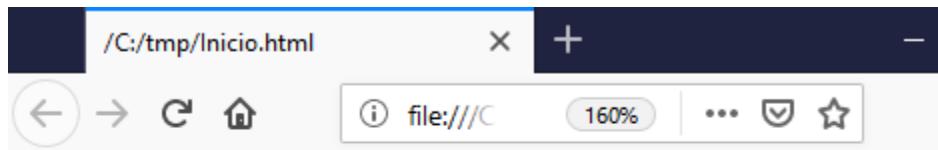


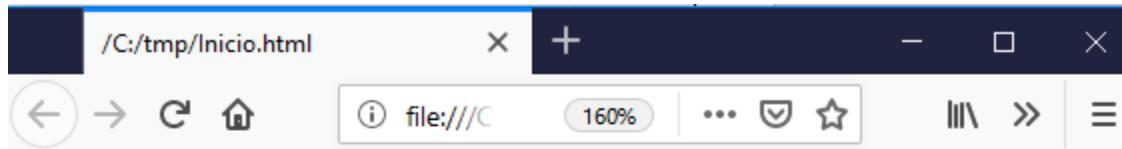
Ilustración 107: Ordenación por el método de burbuja

```

<!DOCTYPE HTML><html><head><script>
//Arreglo de números enteros
var datos = [-363, -1793, -2601, -2541, 6376, 6053, 503, 1601, -2994,
6480, 998];
document.write("Original: " + datos + "<br>");
OrdenarBurbuja(datos);
document.write("Ordenado: " + datos + "<br>");

//Ordenación por burbuja
function OrdenarBurbuja(arreglo){
    for (var i=0; i<arreglo.length-1; i++)
        for (var j=0; j<arreglo.length-i-1; j++)
            if (arreglo[j]>arreglo[j+1]){
                var aux = arreglo[j];
                arreglo[j] = arreglo[j+1];
                arreglo[j+1] = aux;
            }
}
</script></head></html>

```



Original: -363,-1793,-2601,-  
 2541,6376,6053,503,1601,-  
 2994,6480,998  
 Ordenado: -2994,-2601,-2541,-1793,-  
 363,503,998,1601,6053,6376,6480

Ilustración 108: Ordenación por burbuja

```

<!DOCTYPE HTML><html><head><script>
var datos = [-363, -1793, -2601, -2541, 6376, 6053, 503, 1601, -2994];
document.write("Original: " + datos + "<br>");
OrdenarBurbujaMejorado(datos);
document.write("Ordenado: " + datos + "<br>");

//Ordenación por método de burbuja mejorado
function OrdenarBurbujaMejorado(arreglo){
    var intercambio;
    do{
        intercambio = false;
        for (var i = 0; i < arreglo.length - 1; i++) {
            if (arreglo[i] > arreglo[i + 1]) {
                var temporal = arreglo[i];
                arreglo[i] = arreglo[i + 1];
                arreglo[i + 1] = temporal;
                intercambio = true;
            }
        }
    }while (intercambio);
}
</script></head></html>

```

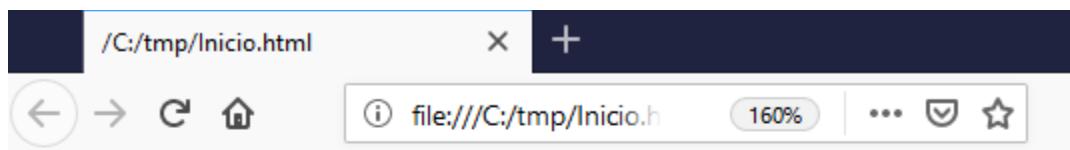


Ilustración 109: Ordenación por método de burbuja mejorado

# Algoritmo de ordenación por selección

Directorio 031

```
<!DOCTYPE HTML><html><head><script>
var letras = ['r', 'v', 'e', 'g', 't', 'u', 'd', 'f', 'f', 'y', 'v'];
document.write("Original: " + letras + "<br>");
OrdenarSeleccion(letras);
document.write("Ordenado: " + letras + "<br>");

//Ordenación por selección
function OrdenarSeleccion(arreglo){
    for(var i=0; i<arreglo.length-1; i++){
        var tmp = i;
        for (var j=i+1; j<arreglo.length; j++)
            if (arreglo[j] < arreglo[tmp])
                tmp = j;

        var temporal = arreglo[tmp];
        arreglo[tmp] = arreglo[i];
        arreglo[i] = temporal;
    }
}
</script></head></html>
```

Directorio 031

```
<!DOCTYPE HTML><html><head><script>
var datos = [-363, -1793, -2601, -2541, 6376, 6053, 503, 1601, -2994,
648];
document.write("Original: " + datos + "<br>");
OrdenarSeleccion(datos);
document.write("Ordenado: " + datos + "<br>");

//Ordenación por selección
function OrdenarSeleccion(arreglo){
    for(var i=0; i<arreglo.length-1; i++){
        var tmp = i;
        for (var j=i+1; j<arreglo.length; j++)
            if (arreglo[j] < arreglo[tmp])
                tmp = j;

        var temporal = arreglo[tmp];
        arreglo[tmp] = arreglo[i];
        arreglo[i] = temporal;
    }
}
</script></head></html>
```

# Algoritmo de ordenación por inserción

Directorio 031

```
<!DOCTYPE HTML><html><head><script>
var letras = ['r', 'v', 'e', 'g', 't', 'u', 'd', 'f', 'f', 'y', 'v', 'm',
'z', 'v'];
document.write("Original: " + letras + "<br>");
OrdenarInsercion(letras);
document.write("Ordenado: " + letras + "<br>");

//Algoritmo de ordenación por inserción
function OrdenarInsercion(arreglo){
    for (var i = 0, j, tmp; i < arreglo.length; ++i) {
        tmp = arreglo[i];
        for (j = i - 1; j >= 0 && arreglo[j] > tmp; --j)
            arreglo[j + 1] = arreglo[j];
        arreglo[j + 1] = tmp;
    }
}
</script></head></html>
```

Directorio 031

```
<!DOCTYPE HTML><html><head><script>
var datos = [-363, -1793, -2601, -2541, 6376, 6053, 503, 1601, -2994,
648];
document.write("Original: " + datos + "<br>");
OrdenarInsercion(datos);
document.write("Ordenado: " + datos + "<br>");

//Algoritmo de ordenación por inserción
function OrdenarInsercion(arreglo){
    for (var i = 0, j, tmp; i < arreglo.length; ++i) {
        tmp = arreglo[i];
        for (j = i - 1; j >= 0 && arreglo[j] > tmp; --j)
            arreglo[j + 1] = arreglo[j];
        arreglo[j + 1] = tmp;
    }
}
</script></head></html>
```

# Algoritmo de ordenación QuickSort

Directorio 031

```
<!DOCTYPE HTML><html><head><script>
var letras = ['r', 'v', 'e', 'g', 't', 'u', 'd', 'f', 'f', 'y', 'v'];
document.write("Original: " + letras + "<br>");
var ordenado = quicksort(letras);
document.write("Ordenado: " + ordenado + "<br>");

//Algoritmo de ordenación QuickSort
function quicksort(arreglo) {
    if (arreglo.length == 0) return [];
    var izquierdo = [], derecho = [], pivote = arreglo[0];
    for (var i = 1; i < arreglo.length; i++)
        if (arreglo[i] < pivote)
            izquierdo.push(arreglo[i]);
        else
            derecho.push(arreglo[i]);

    return quicksort(izquierdo).concat(pivote, quicksort(derecho));
}
</script></head></html>
```

Directorio 031

```
<!DOCTYPE HTML><html><head><script>
var datos = [-363, -1793, -2601, -2541, 6376, 6053, 503, 1601, -2994,
648];
document.write("Original: " + datos + "<br>");
var ordenado = quicksort(datos);
document.write("Ordenado: " + ordenado + "<br>");

function quicksort(arreglo) {
    if (arreglo.length == 0) return [];
    var izquierdo = [], derecho = [], pivote = arreglo[0];
    for (var i = 1; i < arreglo.length; i++)
        if (arreglo[i] < pivote)
            izquierdo.push(arreglo[i]);
        else
            derecho.push(arreglo[i]);

    return quicksort(izquierdo).concat(pivote, quicksort(derecho));
}
</script></head></html>
```

# Algoritmo de ordenación MergeSort

Directorio 031

```
<!DOCTYPE HTML><html><head><script>
var letras = ['r', 'v', 'e', 'g', 't', 'u', 'd', 'f', 'f', 'y', 'v'];
document.write("Original: " + letras + "<br>");
var ordenado = mergesort(letras);
document.write("Ordenado: " + ordenado + "<br>");

function mergesort(arreglo){
    if (arreglo.length <= 1) return arreglo;
    var mitad = Math.floor(arreglo.length / 2);
    var izquierdo = arreglo.slice(0, mitad);
    var derecho = arreglo.slice(mitad, arreglo.length);
    return merge(mergesort(izquierdo), mergesort(derecho));
}

function merge(izquierdo, derecho){
    var ordenado = [];
    while (izquierdo && izquierdo.length > 0 && derecho && derecho.length
> 0) {
        var b = izquierdo[0] <= derecho[0];
        if(b) ordenado.push(izquierdo[0]); else
ordenado.push(derecho[0]);
        if(b) izquierdo.splice(0,1); else derecho.splice(0,1);
    }
    return ordenado.concat(izquierdo, derecho);
}
</script></head></html>
```

```

<!DOCTYPE HTML><html><head><script>
var datos = [-363, -1793, -2601, -2541, 6376, 6053, 503, 1601, -2994,
648];
document.write("Original: " + datos + "<br>");
var ordenado = mergesort(datos);
document.write("Ordenado: " + ordenado + "<br>");

function mergesort(arreglo){
    if (arreglo.length <= 1) return arreglo;
    var mitad = Math.floor(arreglo.length / 2);
    var izquierdo = arreglo.slice(0, mitad);
    var derecho = arreglo.slice(mitad, arreglo.length);
    return merge(mergesort(izquierdo), mergesort(derecho));
}

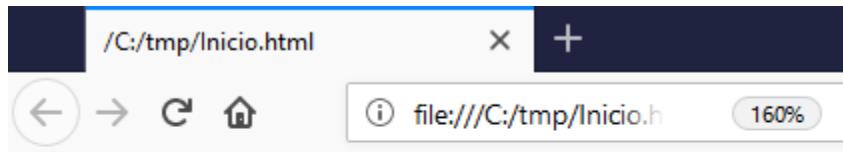
function merge(izquierdo, derecho){
    var ordenado = [];
    while (izquierdo && izquierdo.length > 0 && derecho && derecho.length
> 0) {
        var b = izquierdo[0] <= derecho[0];
        if(b) ordenado.push(izquierdo[0]); else
ordenado.push(derecho[0]);
        if(b) izquierdo.splice(0,1); else derecho.splice(0,1);
    }
    return ordenado.concat(izquierdo, derecho);
}
</script></head></html>

```

## Manejo de Fechas

Directorio 032

```
<!DOCTYPE HTML><html><head><script>
//Mostrar la fecha actual
var fecha = new Date(); //Crea una variable tipo fecha
var dia = fecha.getDate(); //Trae el día
var mes = fecha.getMonth() + 1; //Trae el mes (comienza en 0 para Enero)
var año = fecha.getFullYear(); //Trae el año
document.write("Hoy es " + dia + "/" + mes + "/" + año);
</script></head></html>
```

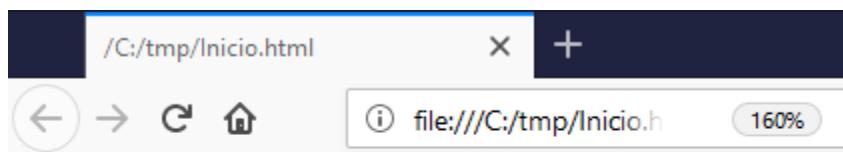


Hoy es 15/1/2019

Ilustración 110: Fecha actual

Directorio 032

```
<!DOCTYPE HTML><html><head><script>
var meses =
["enero","febrero","marzo","abril","mayo","junio","julio","agosto","septiembre",
"octubre","noviembre","diciembre"];
var fecha = new Date();
var dia = fecha.getDate();
var mes = fecha.getMonth();
var año = fecha.getFullYear();
var mesnombre = meses[mes]; //Convierte el valor numérico en nombre de mes
document.write(dia + " de " + mesnombre + " de " + año);
</script></head></html>
```



15 de enero de 2019

Ilustración 111: Fecha actual

```
<!DOCTYPE HTML><html><head><script>
var meses =
["enero", "febrero", "marzo", "abril", "mayo", "junio", "julio", "agosto", "septiembre",
"octubre", "noviembre", "diciembre"];
var diasemana = ["domingo", "lunes", "martes", "miércoles", "jueves",
"viernes", "sábado"];
var fecha = new Date();
var dia = fecha.getDate();
var mes = fecha.getMonth();
var año = fecha.getFullYear();
var diadelasemana = fecha.getDay();
var mesnombre = meses[mes]; //Convierte el valor numérico en nombre de mes
var dianombre = diasemana[diadelasemana]; //Convierte el valor numérico en nombre de día
document.write(dianombre + ", " + dia + " de " + mesnombre + " de " +
año);
</script></head></html>
```

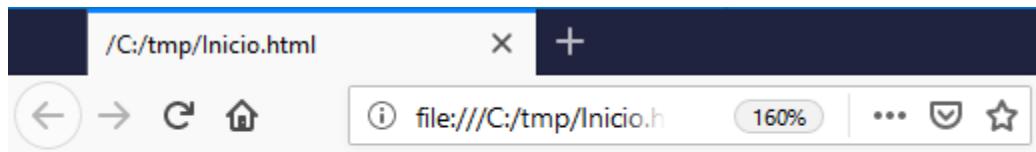
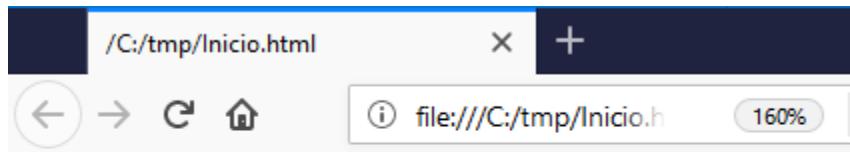


Ilustración 112: Fecha actual

## Manejo de cadenas o strings

Directorio 033

```
<!DOCTYPE HTML><html><head><script>
//Concatenar cadenas
var cadena1 = "Dragón";
var cadena2 = " de ";
var cadena3 = "Komodo";
var cadena4 = cadena1 + cadena2 + cadena3;
document.write(cadena4);
</script></head></html>
```

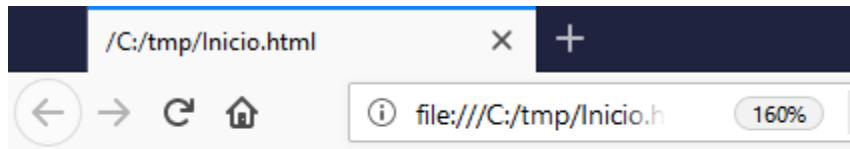


### Dragón de Komodo

Ilustración 113: Concatenación de cadenas

Directorio 033

```
<!DOCTYPE HTML><html><head><script>
//Concatenar cadenas y valores numéricos
var cadena1 = 3;
var cadena2 = "abcde";
var cadena3 = 5;
var cadena4 = cadena1 + cadena2 + cadena3 + "efg";
document.write(cadena4);
</script></head></html>
```



### 3abcde5efg

Ilustración 114: Concatenación de números y cadenas

Directorio 033

```
<!DOCTYPE HTML><html><head><script>
/* Extraer una letra de una cadena. Las cadenas se comportan como arreglos
unidimensionales, el primer elemento está en la posición 0 */
var cadena = "Esta es una prueba";
```

```
var letra = cadena[0]; //Trae la primera letra
document.write(letra);
</script></head></html>
```

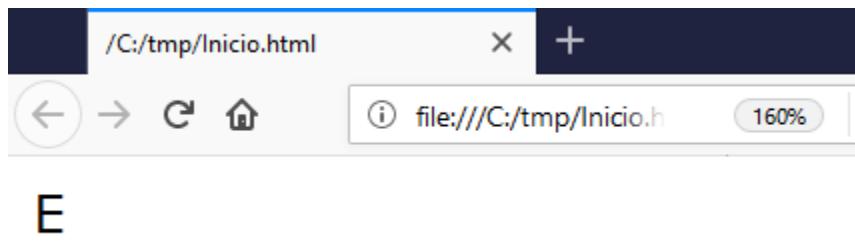


Ilustración 115: Primera letra de una cadena

Directorio 033

```
<!DOCTYPE HTML><html><head><script>
//Longitud de una cadena (incluye los espacios)
var cadena = " Esta es una prueba ";
var longitud = cadena.length;
document.write(longitud);
</script></head></html>
```

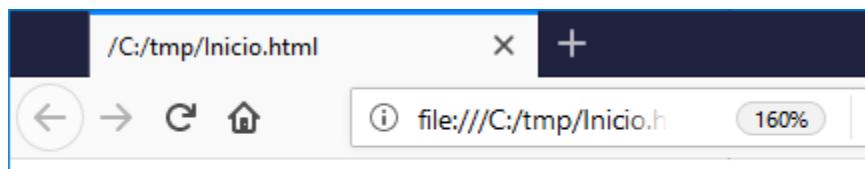


Ilustración 116: Longitud de una cadena

Directorio 033

```
<!DOCTYPE HTML><html><head><script>
//Convertir a mayúsculas y minúsculas
var cadena = "EsTa Es UNa PrUEBa dE CAmbio de MAyúsCULas y MINúscULAs";
var mayuscula = cadena.toUpperCase(); //Convierte a mayúsculas
var minuscula = cadena.toLowerCase(); //Convierte a minúsculas
document.write(mayuscula);
document.write("<br>");
document.write(minuscula);
</script></head></html>
```



Ilustración 117: Convertir a mayúsculas y minúsculas

Directorio 033

```
<!DOCTYPE HTML><html><head><script>
//Comparación de cadenas
var cadena1 = "FraseA";
var cadena2 = "FraseA";
if (cadena1 === cadena2) //Comparación estricta al usar ===
    document.write("iguales");
else
    document.write("diferentes");
</script></head></html>
```

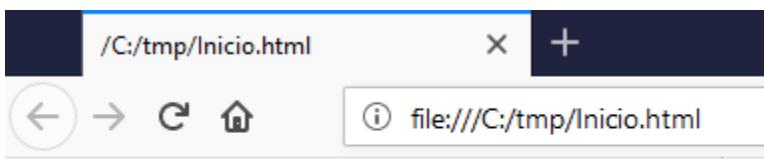


Ilustración 118: Comparación de cadenas

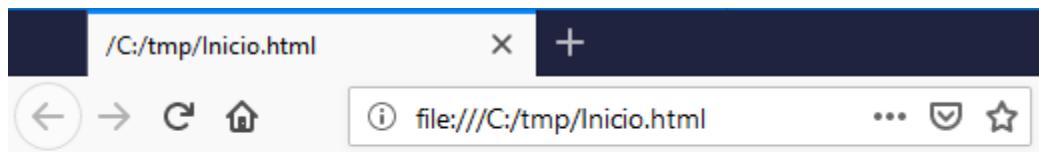
## Ejemplos de programas usando cadenas

### Invertir Cadena

Directorio 034

```
<!DOCTYPE HTML><html><head><script>
//Invierte la cadena
var cadena = "Rafael Alberto Moreno Parra";
var nueva = "";
for (var cont=0; cont<cadena.length; cont++) nueva = cadena[cont] + nueva;

document.write("Original: " + cadena + "<br>");
document.write("Invierte: " + nueva + "<br>");
</script></head></html>
```



Original: Rafael Alberto Moreno Parra  
Invierte: arraP oneroM otreblA leafaR

Ilustración 119: Invierte la cadena

## Quitar los espacios de una cadena

Directorio 034

```
<!DOCTYPE HTML><html><head><script>
//Quita los espacios de una cadena
var cadena = " a b c d e f ";
var nueva = "";
for (var cont=0; cont<cadena.length; cont++)
    if (cadena[cont] != ' ')
        nueva += cadena[cont];

document.write("Original: [" + cadena + "]<br>");
document.write("Sin espacios : [" + nueva + "]<br>"); 
</script></head></html>
```



Ilustración 120: Quita los espacios de una cadena

## Quitar las vocales de una cadena

Directorio 034

```
<!DOCTYPE HTML><html><head><script>
//Quitar vocales (minúsculas, mayúsculas, tildadas)
var cadena = prompt("Escriba un texto");
var nueva = "";
for (var cont=0; cont<cadena.length; cont++)
    if (!EsVocal(cadena[cont])) nueva += cadena[cont];
document.write("Original: " + cadena + "<br>");
document.write("Sin vocales : " + nueva + "<br>");

function EsVocal(caracter){ //Retorna true si el caracter es una vocal
    var vocales = "aeiouAEIOUÁÉÍÓÚáéíóúäëöü";
    for (var letra=0; letra < vocales.length; letra++)
        if (caracter == vocales[letra])
            return true;

    return false;
}
</script></head></html>
```

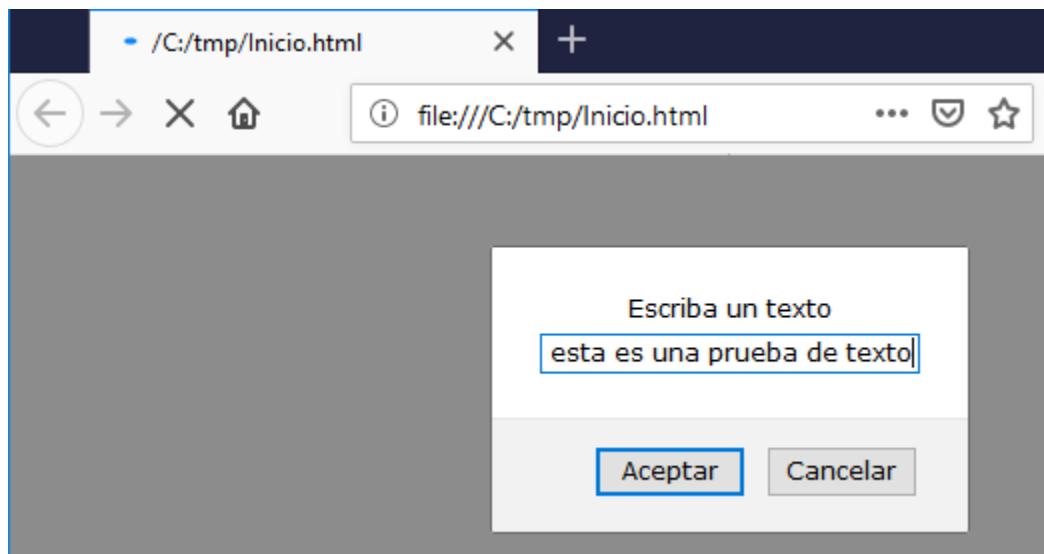


Ilustración 121: Quitar vocales (minúsculas, mayúsculas, tildadas)

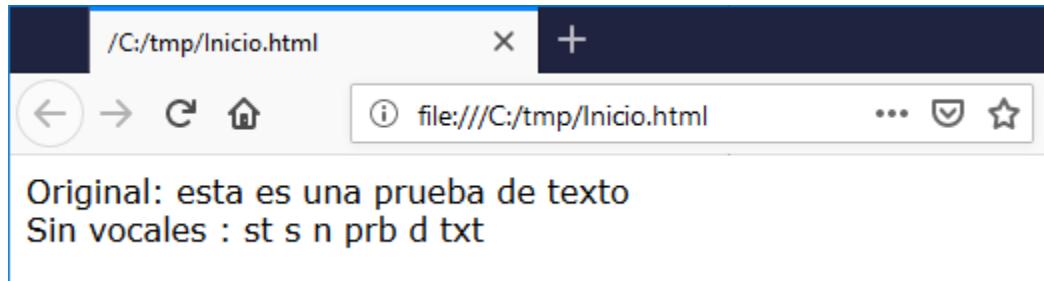


Ilustración 122: Quitar vocales (minúsculas, mayúsculas, tildadas)

## Quitar caracteres no permitidos

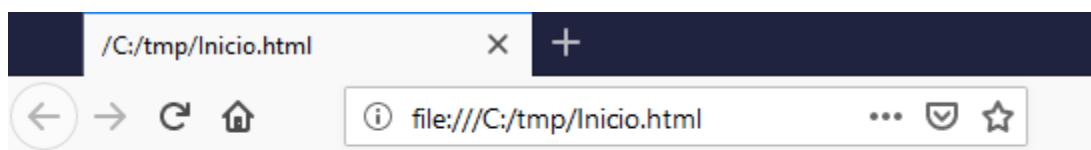
Directorio 034

```
<!DOCTYPE HTML><html><head><script>
//Retirar caracteres no permitidos de una cadena
var cadena = "Esto es una <html><big>prueba</big></html>";
var nueva = "";
for (var cont=0; cont<cadena.length; cont++)
    if (EsPermitido(cadena[cont])) nueva += cadena[cont];

document.write("Original: " + cadena + "<br>");
document.write("Con los caracteres permitidos: " + nueva + "<br>");

//Retorna true si el caracter está permitido
function EsPermitido(caracter){
    var permite = "ÁÉÍÓÚáéíóúäëöü";
    permite += "abcdefghijklmnopqrstuvwxyz";
    permite += "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    permite += " 1234567890";
    for (var letra=0; letra < permite.length; letra++)
        if (caracter == permite[letra])
            return true;

    return false;
}
</script></head></html>
```



Original: Esto es una prueba  
Con los caracteres permitidos: Esto es una htmlbigpruebabightml

Ilustración 123: Retirar caracteres no permitidos de una cadena

## Un sencillo cifrado / descifrado

Directorio 034

```
<!DOCTYPE HTML><html><head><script>
//Cifrado sencillo
var cadena = prompt("Escriba un texto");
var cifrada = Cifrar(cadena);
document.write("Original: " + cadena + "<br>");
document.write("Cifrado: " + cifrada + "<br>");

var descifra = DesCifrar(cifrada);
document.write("Descifrado: " + descifra + "<br>");

//Cifra la cadena moviendo un carácter hacia adelante
function Cifrar(cadena){
    var cifrada = "";
    for (var cont=0; cont<cadena.length; cont++) {
        var num = cadena[cont].charCodeAt(); //Trae el valor ASCII de la
letra
        num++; //Incrementa en uno ese valor
        cifrada += String.fromCharCode(num); //Convierte el valor a su
respectivo carácter
    }
    return cifrada;
}

//DesCifra la cadena cifrada moviendo un carácter hacia atrás
function DesCifrar(cadena){
    var descifrada = "";
    for (var cont=0; cont<cadena.length; cont++) {
        var num = cadena[cont].charCodeAt(); //Trae el valor ASCII de la
letra
        num--; //Incrementa en uno ese valor
        descifrada += String.fromCharCode(num); //Convierte el valor a su
respectivo carácter
    }
    return descifrada;
}
</script></head></html>
```

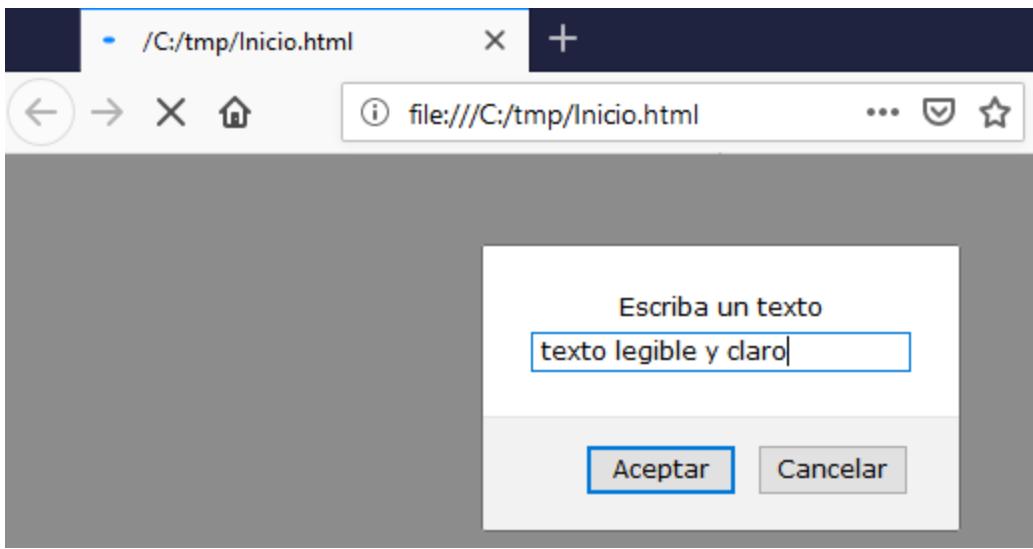


Ilustración 124: Cifrado sencillo

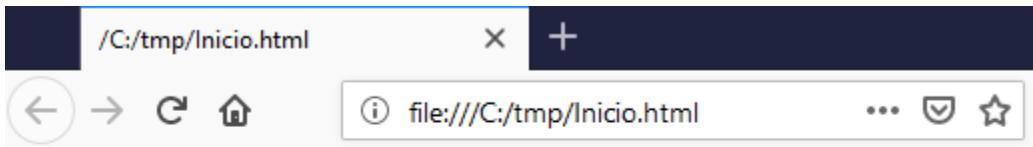


Ilustración 125: Cifrado sencillo

## Un sencillo cifrado / descifrado con clave de cifrado

Directorio 034

```
<!DOCTYPE HTML><html><head><script>
//Cifrado sencillo con clave de cifrado
var cadena = prompt("Escriba texto");
var clave = prompt("Clave de cifrado");

var cifrada = Cifrar(cadena, clave);
document.write("Original: " + cadena + "<br>");
document.write("Cifrado: " + cifrada + "<br>");

var descifra = DesCifrar(cifrada, clave);
document.write("Descifrado: " + descifra + "<br>");

function Cifrar(cadena, clave){
    var cifrada = "";
    var clavecont = 0;
    for (var cont=0; cont<cadena.length; cont++) {
        //Cada letra de la clave genera un desplazamiento
        desplaza = clave[clavecont++].charCodeAt()%20;
        if (clavecont >= clave.length) clavecont = 0; //Si se llega al
final de la clave
        cifrada += String.fromCharCode(cadena[cont].charCodeAt() + desplaza);
    }
    return cifrada;
}

function DesCifrar(cadena, clave){
    var descifrada = "";
    var clavecont = 0;
    for (var cont=0; cont<cadena.length; cont++) {
        //Cada letra de la clave genera un desplazamiento
        desplaza = clave[clavecont++].charCodeAt()%20;
        if (clavecont >= clave.length) clavecont = 0; //Si se llega al
final de la clave
        descifrada += String.fromCharCode(cadena[cont].charCodeAt() - desplaza);
    }
    return descifrada;
}
</script></head></html>
```

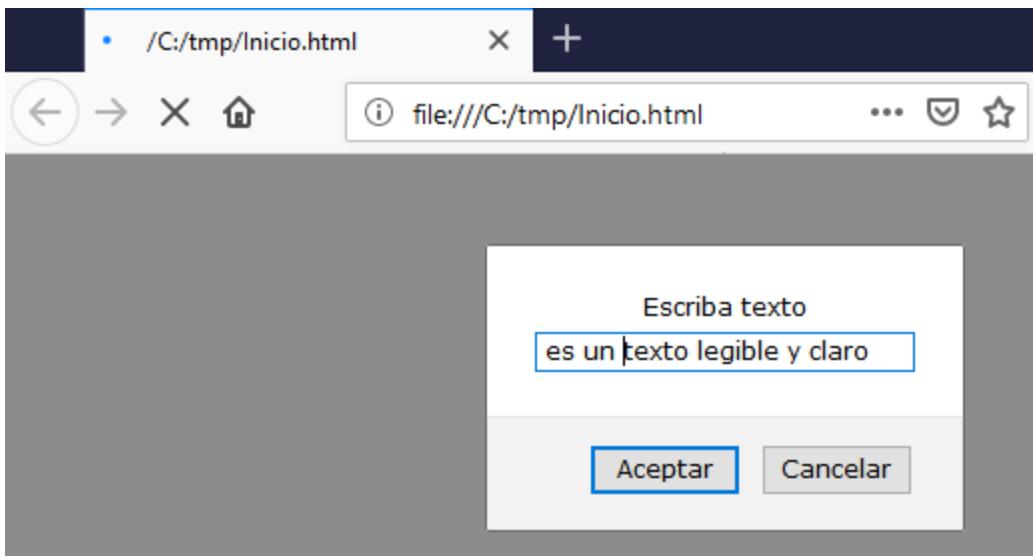


Ilustración 126: Cifrado sencillo con clave de cifrado

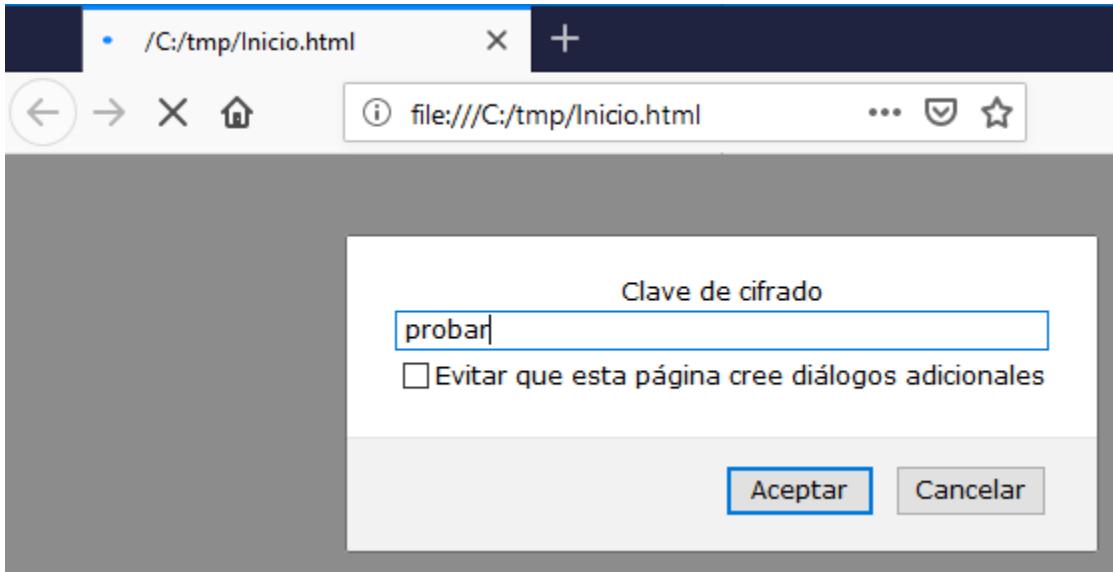


Ilustración 127: Cifrado sencillo con clave de cifrado

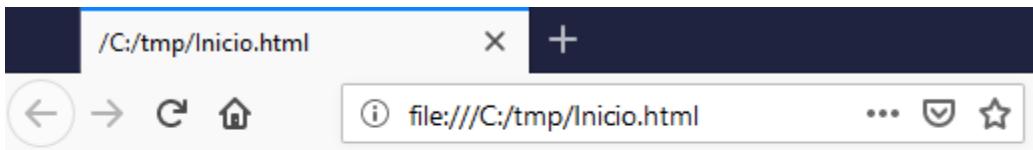


Ilustración 128: Cifrado sencillo con clave de cifrado

## Sumar largos números enteros almacenados en cadenas

Directorio 034

```
<!DOCTYPE HTML><html><head><script> //Suma dos números enteros positivos
almacenados en cadenas
var numeroA = "395763487398655335"; //Primer número
var numeroB = "8888888888888888"; //Segundo número
var resultado = ""; //Guarda el resultado
var ultimaPosA = numeroA.length - 1; //Posición del último dígito del
primer número
var ultimaPosB = numeroB.length - 1;
var llevar = false; //Se activa si la suma de dígitos es 10 o más
significativo al más significativo
var suma = 0; //Suma de dígitos
if (ultimaPosA>=0 && ultimaPosB>=0) //Si ambos dígitos existen en
esas posiciones
    suma = parseInt(numeroA[ultimaPosA]) +
parseInt(numeroB[ultimaPosB]);
else if (ultimaPosA>=0) //Si sólo existe dígito en primer número
    suma = parseInt(numeroA[ultimaPosA]);
else //sólo existe dígito en segundo número
    suma = parseInt(numeroB[ultimaPosB]);
if (llevar==true) suma++; //Si llevaba aumenta en 1 la suma
llevar = false; //La bandera de llevar se apaga
if (suma>=10){ llevar=true; suma-=10; } //Si la suma es 10 o más,
entonces lleva, le quita 10
resultado = suma + resultado; //Agrega al inicio de la cadena
ultimaPosA--; //Va hacia la izquierda del primer número
ultimaPosB--; //Va hacia la izquierda del segundo número
}
if (llevar==true) resultado = "1" + resultado; //Si la suma de los
primeros dígitos es 10 o más
document.write(numeroA + " + <br>" + numeroB + "<br>-----"-
---<br>" + resultado);
</script></head><body></body></html>
```

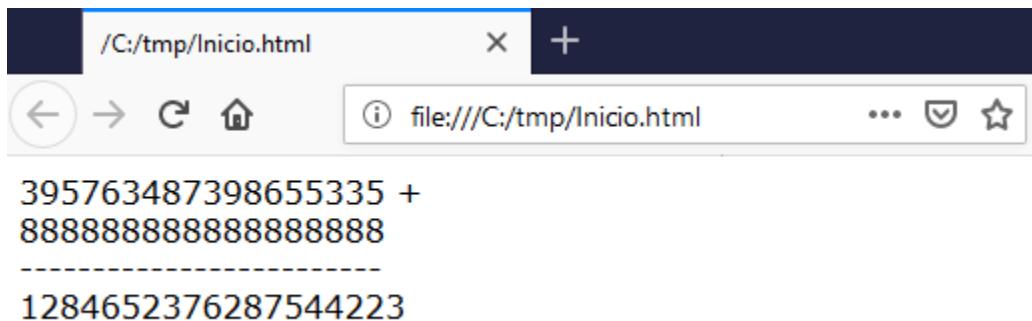


Ilustración 129: Suma dos números enteros positivos almacenados en cadenas

## Arreglos bidimensionales

Directorio 035

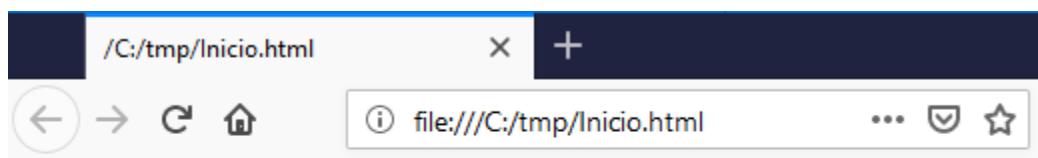
```
<!DOCTYPE HTML><html><head><script>
//Crea el arreglo bidimensional
var bidiarreglo = new Array(2); //Crea dos filas

//Crea cuatro columnas por fila
for (var fila=0; fila < bidiarreglo.length; fila++) {
    bidiarreglo[fila] = new Array(4);

//Tenemos un arreglo bidimensional de 2 filas * 4 columnas.
//Las coordenadas son [fila][columna]. Empiezan en [0,0] y terminan en
[1,3]
    bidiarreglo[0][0] = 11;
    bidiarreglo[0][1] = 13;
    bidiarreglo[0][2] = 15;
    bidiarreglo[0][3] = 17;

    bidiarreglo[1][0] = 21;
    bidiarreglo[1][1] = 23;
    bidiarreglo[1][2] = 25;
    bidiarreglo[1][3] = 27;

//Imprimir el arreglo bidimensional
for (var fila=0; fila < bidiarreglo.length; fila++) {
    document.write("<br>");
    for (var columna=0; columna < bidiarreglo[fila].length; columna++)
        document.write(bidiarreglo[fila][columna] + " , ");
}
</script></head></html>
```



11 , 13 , 15 , 17 ,  
21 , 23 , 25 , 27 ,

Ilustración 130: Crea el arreglo bidimensional

Directorio 035

```
<!DOCTYPE HTML><html><head><script>
//Crea el arreglo bidimensional
var tabla = new Array(8); //Crea las filas
```

```

//Crea las columnas
for (var fila=0; fila < tabla.length; fila++)
    tabla[fila] = new Array(20);

//Llena el arreglo bidimensional de puntos
for (var fila=0; fila < tabla.length; fila++)
    for (var columna=0; columna < tabla[fila].length; columna++)
        tabla[fila][columna]='.';

//Pone una X en una posición al azar
var posF = Math.floor(Math.random()*8);
var posC = Math.floor(Math.random()*20);
tabla[posF][posC] = 'X';

//Imprime la tabla
for (var fila=0; fila < tabla.length; fila++) {
    document.write("<br>");
    for (var columna=0; columna < tabla[fila].length; columna++)
        document.write(tabla[fila][columna]);
}
</script></head></html>

```

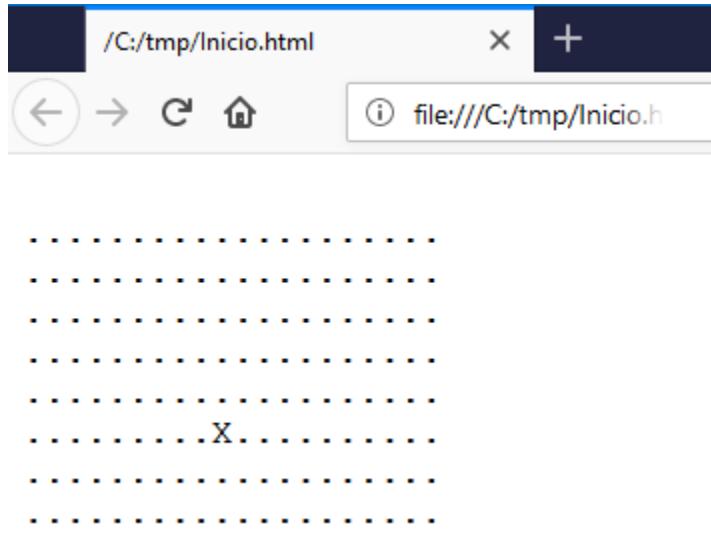


Ilustración 131: Pone una X en una posición al azar

```

<!DOCTYPE HTML><html><head><script>
//Llama a función que genera arreglo bidimensional
var tabla = GenerarArregloBidimensional(8);

//Llena el arreglo bidimensional de puntos
for (var fila=0; fila < tabla.length; fila++)
    for (var columna=0; columna < 20; columna++)
        tabla[fila][columna]='.';

//Pone una X en una posición al azar
var posF = Math.floor(Math.random()*8);
var posC = Math.floor(Math.random()*20);
tabla[posF][posC] = 'X';

//Imprime la tabla
for (var fila=0; fila < tabla.length; fila++) {
    document.write("<br>");
    for (var columna=0; columna < tabla[fila].length; columna++)
        document.write(tabla[fila][columna]);
}

//Función genérica para crear arreglo bidimensional
function GenerarArregloBidimensional(totalfilas) {
    var arreglo = [];
    for (var fila=0; fila < totalfilas; fila++) arreglo[fila] = [];
    return arreglo;
}
</script></head></html>

```

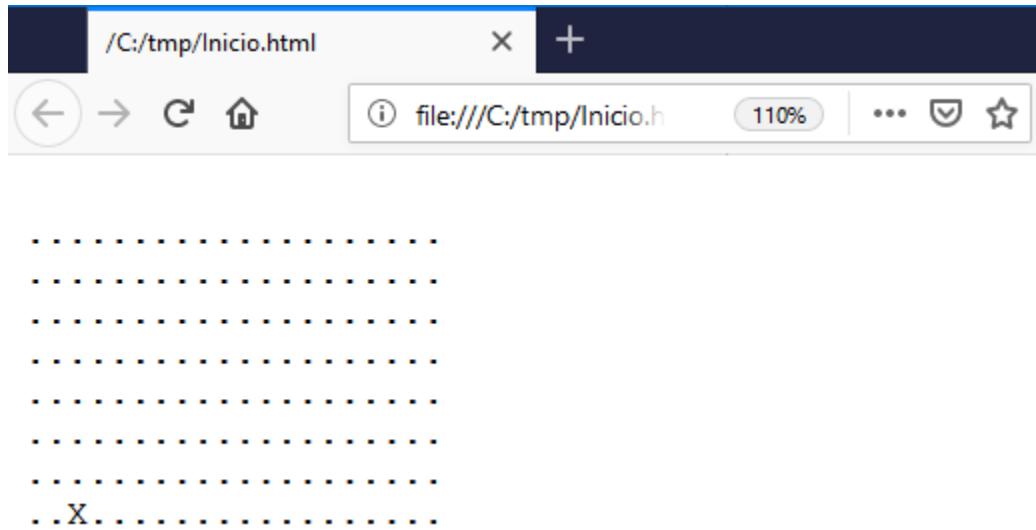


Ilustración 132: Función genérica para crear arreglo bidimensional

```

<!DOCTYPE HTML><html><head><script>

var tabla = GeneraArregloBidimensional(10, 15, '.');
ImprimeArregloBidimensional(tabla);

/* Función genérica para crear arreglos bidimensionales
   con el número de filas y columnas dado y llena cada celda
   de un valor */
function GeneraArregloBidimensional(filas, columnas, valor){
    var arreglo = [];
    for (var fila=0; fila<filas; fila++) {
        arreglo[fila] = [];
        for (var columna=0; columna<columnas; columna++)
            arreglo[fila][columna]=valor;
    }
    return arreglo;
}

//Función genérica para imprimir un arreglo bidimensional
function ImprimeArregloBidimensional(arreglo){
    for (var fila=0; fila < arreglo.length; fila++){
        document.write("<br>");
        for (var columna=0; columna < arreglo[fila].length; columna++)
            document.write(arreglo[fila][columna]);
    }
}
</script></head></html>

```

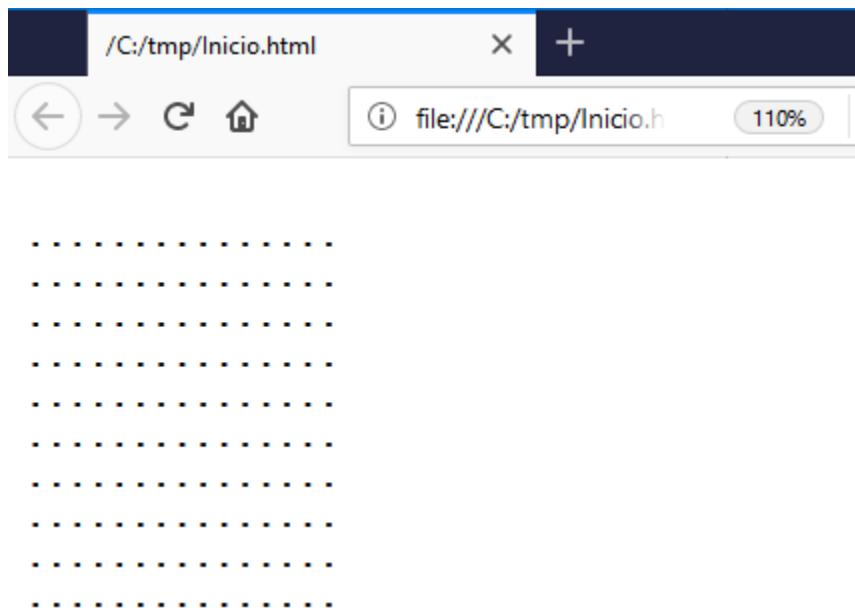


Ilustración 133: Función genérica para crear arreglo bidimensional

## Ejemplos de programas usando arreglos bidimensionales

### Poner una torre al azar en un tablero de ajedrez y mostrar su ataque

Directorio 036

```
<!DOCTYPE HTML><html><head><script>
/* Crear un tablero de ajedrez de 8*8, poner en una posición al
azar una Torre y mostrar su ataque */

var tabla = GeneraArregloBidimensional(8, 8, '.');
PoneTorreAzar(tabla);
ImprimeArregloBidimensional(tabla);

//Pone una torre al azar y muestra su ataque
function PoneTorreAzar(arreglo){
    var filaAzar = Math.floor(Math.random()*arreglo.length);
    var coluAzar = Math.floor(Math.random()*arreglo[0].length);
    for (var fil=0; fil<arreglo.length; fil++) arreglo[fil][coluAzar] =
'x';
    for (var col=0; col<arreglo[0].length; col++) arreglo[filaAzar][col] =
'x';
    arreglo[filaAzar][coluAzar] = 'T';
}

function GeneraArregloBidimensional(filas, columnas, valor){
    var arreglo = [];
    for (var fila=0; fila<filas; fila++){
        arreglo[fila] = [];
        for (var columna=0; columna<columnas; columna++)
            arreglo[fila][columna]=valor;
    }
    return arreglo;
}
function ImprimeArregloBidimensional(arreglo){
    for (var fila=0; fila < arreglo.length; fila++){
        document.write("<p style='font-family:courier new;'>");
        for (var columna=0; columna < arreglo[fila].length; columna++)
            document.write(arreglo[fila][columna]);
        document.write("</p>");
    }
}
</script></head></html>
```

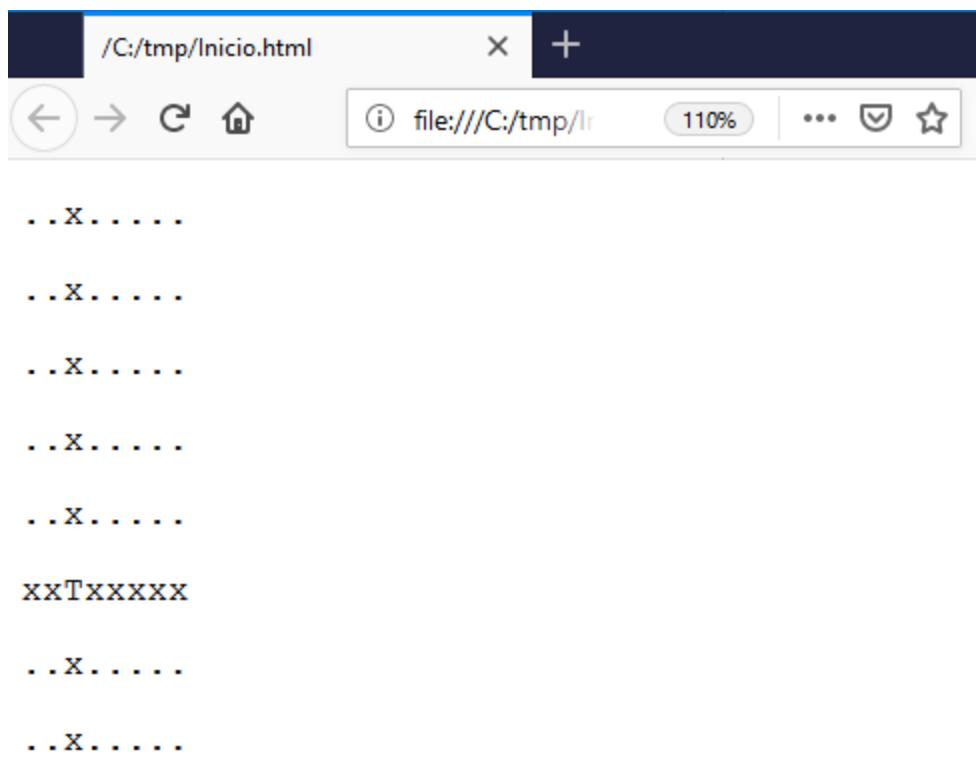


Ilustración 134: Poner en una posición al azar la torre y mostrar su ataque

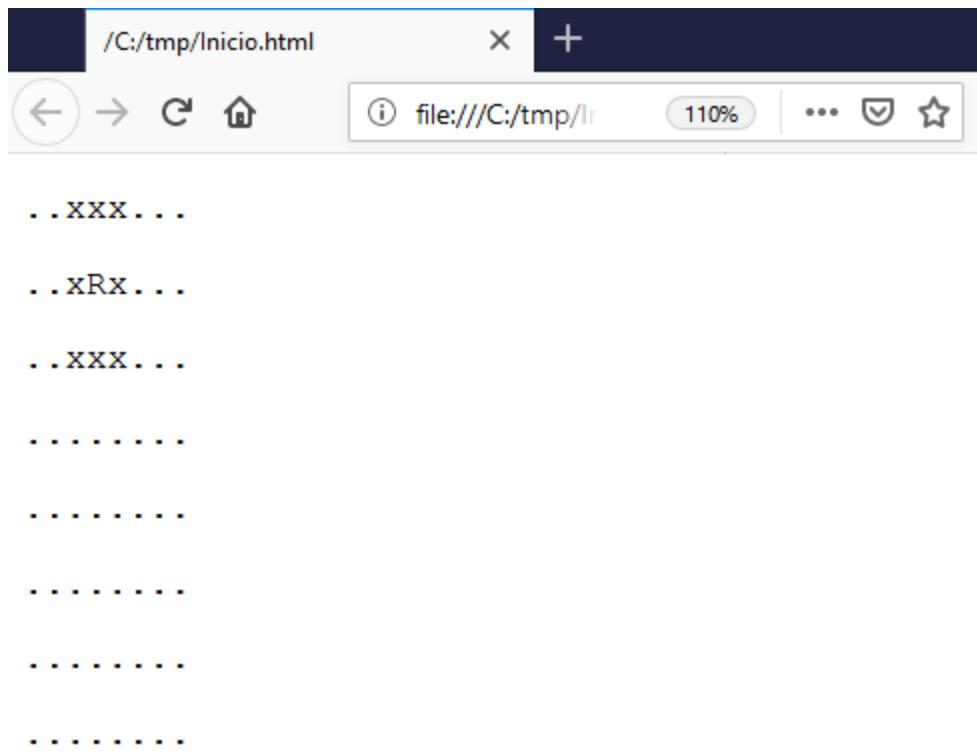
# Poner un Rey al azar en un tablero de ajedrez y mostrar su ataque

Directorio 036

```
<!DOCTYPE HTML><html><head><script>
/* Crear un tablero de ajedrez de 8*8, poner en una posición al
azar el Rey y mostrar su ataque */
var tabla = GeneraArregloBidimensional(8, 8, '.');
PoneReyAzar(tabla);
ImprimeArregloBidimensional(tabla);

//Pone un Rey al azar y muestra su ataque
function PoneReyAzar(arreglo){
    var filaAzar = Math.floor(Math.random()*arreglo.length);
    var coluAzar = Math.floor(Math.random()*arreglo[0].length);
    if (filaAzar>0) arreglo[filaAzar-1][coluAzar]='x';
    if (filaAzar>0 && coluAzar>0) arreglo[filaAzar-1][coluAzar-1]='x';
    if (coluAzar>0) arreglo[filaAzar][coluAzar-1]='x';
    if (coluAzar<arreglo[0].length-1) arreglo[filaAzar][coluAzar+1]='x';
    if (filaAzar>0 && coluAzar<arreglo[0].length-1) arreglo[filaAzar-1][coluAzar+1]='x';
    if (filaAzar<arreglo.length-1 && coluAzar<arreglo[0].length-1)
arreglo[filaAzar+1][coluAzar+1]='x';
    if (filaAzar<arreglo.length-1) arreglo[filaAzar+1][coluAzar]='x';
    if (filaAzar<arreglo.length-1 && coluAzar>0)
arreglo[filaAzar+1][coluAzar-1]='x';
    arreglo[filaAzar][coluAzar] = 'R';
}

function GeneraArregloBidimensional(filas, columnas, valor){
    var arreglo = [];
    for (var fila=0; fila<filas; fila++){
        arreglo[fila] = [];
        for (var columna=0; columna<columnas; columna++)
            arreglo[fila][columna]=valor;
    }
    return arreglo;
}
function ImprimeArregloBidimensional(arreglo){
    for (var fila=0; fila < arreglo.length; fila++){
        document.write("<p style='font-family:courier new;'>");
        for (var columna=0; columna < arreglo[fila].length; columna++)
            document.write(arreglo[fila][columna]);
        document.write("</p>");
    }
}
</script></head></html>
```



*Ilustración 135: Poner un Rey al azar en un tablero de ajedrez y mostrar su ataque*

# Poner un alfil al azar en un tablero de ajedrez y mostrar su ataque

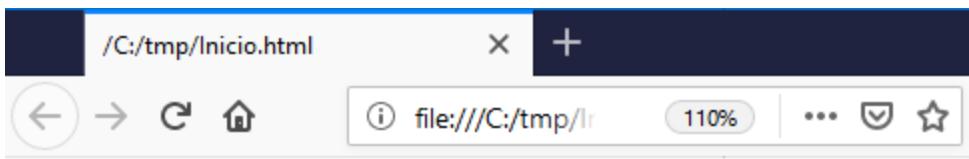
Directorio 036

```
<!DOCTYPE HTML><html><head><script>
/* Crear un tablero de ajedrez de 8*8, poner en una posición al
azar un Alfil y mostrar su ataque */
var tabla = GeneraArregloBidimensional(8, 8, '.');
PoneAlfilAzar(tabla);
ImprimeArregloBidimensional(tabla);

function PoneAlfilAzar(arreglo){
    var filaAzar = Math.floor(Math.random()*arreglo.length);
    var coluAzar = Math.floor(Math.random()*arreglo[0].length);
    var tmpFil=filaAzar;
    var tmpCol=coluAzar;
    while(tmpFil>=0 && tmpCol>=0) { arreglo[tmpFil][tmpCol]='x'; tmpFil--;
    ; tmpCol--; }
    tmpFil=filaAzar; tmpCol=coluAzar;
    while(tmpFil>=0 && tmpCol<8) { arreglo[tmpFil][tmpCol]='x'; tmpFil--;
    ; tmpCol++; }
    tmpFil=filaAzar; tmpCol=coluAzar;
    while(tmpFil<8 && tmpCol<8) { arreglo[tmpFil][tmpCol]='x';
    ; tmpFil++; tmpCol++; }
    tmpFil=filaAzar; tmpCol=coluAzar;
    while(tmpFil<8 && tmpCol>=0) { arreglo[tmpFil][tmpCol]='x';
    ; tmpFil++; tmpCol--; }
    arreglo[filaAzar][coluAzar] = 'A';
}

function GeneraArregloBidimensional(filas, columnas, valor){
    var arreglo = [];
    for (var fila=0; fila<filas; fila++){
        arreglo[fila] = [];
        for (var columna=0; columna<columnas; columna++)
            arreglo[fila][columna]=valor;
    }
    return arreglo;
}

function ImprimeArregloBidimensional(arreglo){
    for (var fila=0; fila < arreglo.length; fila++){
        document.write("<p style='font-family:courier new;'>");
        for (var columna=0; columna < arreglo[fila].length; columna++)
            document.write(arreglo[fila][columna]);
        document.write("</p>");
    }
}
</script></head></html>
```



.....

.....X

X.....X.

.X...X..

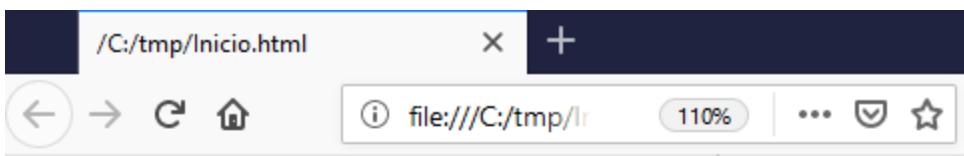
..X.X...

...A....

..X.X...

.X...X..

Ilustración 136: Poner un alfil al azar en un tablero de ajedrez y mostrar su ataque



...X.X..

....A...

..X.X..

..X...X.

.X.....X

X.....

.....

.....

Ilustración 137: Poner un alfil al azar en un tablero de ajedrez y mostrar su ataque

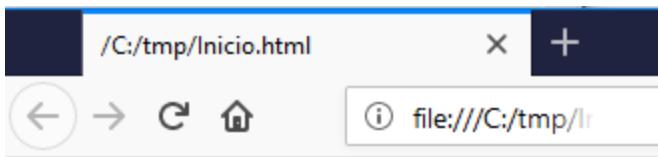
# Poner una Reina al azar en un tablero de ajedrez y mostrar su ataque

Directorio 036

```
<!DOCTYPE HTML><html><head><script>
/* Crear un tablero de ajedrez de 8*8, poner en una posición al
azar la Reina y mostrar su ataque */
var tabla = GeneraArregloBidimensional(8, 8, '.');
PoneReinaAzar(tabla);
ImprimeArregloBidimensional(tabla);

function PoneReinaAzar(arreglo){
    var filaAzar = Math.floor(Math.random()*arreglo.length);
    var coluAzar = Math.floor(Math.random()*arreglo[0].length);
    var tmpFil=filaAzar;
    var tmpCol=coluAzar;
    while(tmpFil>=0 && tmpCol>=0) { arreglo[tmpFil][tmpCol]='x'; tmpFil--;
; tmpCol--; }
    tmpFil=filaAzar; tmpCol=coluAzar;
    while(tmpFil>=0 && tmpCol<8) { arreglo[tmpFil][tmpCol]='x'; tmpFil--;
; tmpCol++; }
    tmpFil=filaAzar; tmpCol=coluAzar;
    while(tmpFil<8 && tmpCol<8) { arreglo[tmpFil][tmpCol]='x'; tmpFil++;
; tmpCol++; }
    tmpFil=filaAzar; tmpCol=coluAzar;
    while(tmpFil<8 && tmpCol>=0) { arreglo[tmpFil][tmpCol]='x';
tmpFil++; tmpCol--; }
    for (tmpFil=0; tmpFil<8; tmpFil++) arreglo[tmpFil][coluAzar]='x';
    for (tmpCol=0; tmpCol<8; tmpCol++) arreglo[filaAzar][tmpCol]='x';
    arreglo[filaAzar][coluAzar] = 'R';
}

function GeneraArregloBidimensional(filas, columnas, valor){
    var arreglo = [];
    for (var fila=0; fila<filas; fila++){
        arreglo[fila] = [];
        for (var columna=0; columna<columnas; columna++)
            arreglo[fila][columna]=valor;
    }
    return arreglo;
}
function ImprimeArregloBidimensional(arreglo){
    for (var fila=0; fila < arreglo.length; fila++){
        document.write("<p style='font-family:courier new;'>");
        for (var columna=0; columna < arreglo[fila].length; columna++)
            document.write(arreglo[fila][columna]);
        document.write("</p>");
    }
}
</script></head></html>
```



....X....

X...X....

.X..X..X

..X.X.X.

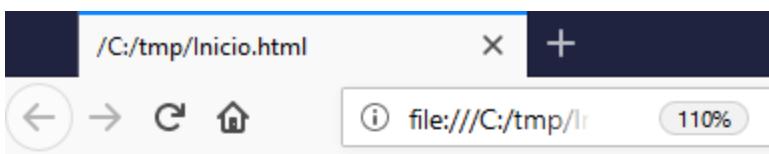
...XXX..

xxxxRxxx

...XXX..

..X.X.X.

Ilustración 138: Poner una Reina al azar en un tablero de ajedrez y mostrar su ataque



....X....

...X....

...X....

...X...X

X..X..X.

.X..X..

...XXX..

xxxRxxxx

Ilustración 139: Poner una Reina al azar en un tablero de ajedrez y mostrar su ataque

## Ruta del menor costo

Dada una tabla (arreglo bidimensional) de costos que muestra cuánto cuesta ir de una ciudad a otra. El reto es encontrar una ruta que visite todas las ciudades, sin repetir ninguna y que tenga el menor coste. Si nos ponemos sistemáticos habría que probar  $N!$  ( $N$  factorial) rutas para encontrar la de menor costo, un proceso muy costoso computacionalmente y no terminaría en un tiempo justo. El método mostrado es usando una técnica indeterminista que se acerca a la ruta menos costosa.

Directorio 036

```
<!DOCTYPE HTML><html><head><script>
/* Hay N ciudades a recorrer (0 a N-1). Sólo se puede visitar una ciudad
por vez.
En la tabla aparece cuánto cuesta ir de una ciudad (origen) a otra
ciudad (destino).
    ¿Qué ruta tomar para visitar todas las ciudades con el mínimo costo?
*/
var ciudad = 20; //Número de ciudades
var minValor=15; //Valor mínimo que tendrá ir de una ciudad a otra
var maxValor=85; //Valor máximo que tendrá ir de una ciudad a otra

var dest1, dest2; //Ciudad origen a Ciudad destino

//Genera valores de viaje al azar
var valorviajes = inicializavalorviajes(ciudad, minValor, maxValor);

//Imprime los valores
imprime(valorviajes);

//Inicia con una ruta predeterminada 0, 1, 2, 3, .... N ... 0
var ruta = iniciaRuta(ciudad);

//Deduce el costo de esa ruta predeterminada
var costo = DeduceCosto(ruta, valorviajes);
document.write("<br>" + ruta + "=>" + costo);

//Usando el método Monte Carlo se buscarán otras rutas con menor costo
for(var pruebas=1; pruebas<=50000; pruebas++){
    dest1 = Math.floor(Math.random()*ruta.length);
    do{
        dest2 = Math.floor(Math.random()*ruta.length);
    }while (dest2==dest1)
    ModificaRuta(ruta, dest1, dest2)
    var costoNuevo = DeduceCosto(ruta, valorviajes);
    if (costoNuevo < costo){
        costo = costoNuevo;
        document.write("<br>" + ruta + "=>" + costo);
    }
    else
        ModificaRuta(ruta, dest1, dest2); //Dejar la ruta como antes
}
```

```

}

//Modifica la ruta de viaje
function ModificaRuta(ruta, dest1, dest2){
    var tmp = ruta[dest1];
    ruta[dest1]=ruta[dest2];
    ruta[dest2]=tmp;
}

//Inicia el arreglo bidimensional de rutas
function iniciaRuta(limite){
    var ruta = new Array();
    for(var cont=0; cont<limite; cont++) ruta.push(cont);
    return ruta;
}

//Deduce el costo de la ruta de viaje
function DeducCosto(ruta, costos){
    var acum=0;
    for (var cont=0; cont<ruta.length-1; cont++)
        acum += costos[ruta[cont]][ruta[cont+1]];
    return acum;
}

//Llena de valores al azar la tabla de costos de viajes de una ciudad a
//otra
function inicializavalorviajes(ciudad, minValor, maxValor){
    var tablero = new Array();
    for (var fila=0; fila<ciudad; fila++){
        tablero[fila] = [];
        for (var columna=0; columna<ciudad; columna++)
            tablero[fila][columna]=Math.floor(Math.random()*(maxValor-
minValor)+minValor);
    }
    for (var cont=0; cont<ciudad; cont++) tablero[cont][cont]=0;
    return tablero;
}

//Imprime el tablero
function imprime(tablero){
    for (var fila=0; fila<tablero.length; fila++) {
        for (var col=0; col<tablero.length; col++)
            document.write(tablero[fila][col] + " ");
        document.write("<br>");
    }
}

</script></head></html>

```

*Ilustración 140: Ruta del menor costo. Genera nuevas rutas más económicas*

# Resolver Sudokus

Directorio 036

```
<!DOCTYPE HTML><html><head><script>

var sudoku = /* El Sudoku como es planteado. Los ceros(0) son las casillas vacías a completar */
[[5, 3, 0, 0, 7, 0, 0, 0, 0],
 [6, 0, 0, 1, 9, 5, 0, 0, 0],
 [0, 9, 8, 0, 0, 0, 0, 6, 0],
 [8, 0, 0, 0, 6, 0, 0, 0, 3],
 [4, 0, 0, 8, 0, 3, 0, 0, 1],
 [7, 0, 0, 0, 2, 0, 0, 0, 6],
 [0, 6, 0, 0, 0, 0, 2, 8, 0],
 [0, 0, 0, 4, 1, 9, 0, 0, 5],
 [0, 0, 0, 0, 8, 0, 0, 7, 9]
];

var finalizo=false; /* Mantiene el ciclo hasta que resuelva el Sudoku */
var ciclos=0; /* Lleva el número de iteraciones */

var solucion = new Array(); /* Tablero en el que se trabaja */
for (var fila=0; fila<9; fila++){
    solucion[fila] = [];
    for (var columna=0; columna<9; columna++)
        solucion[fila][columna]=0;
}
var DESTRUYE = 700; /* Cada cuantos ciclos borra números para destrabar */
*/

/* Ciclo que llenará el sudoku completamente */
do{
    for (var fila=0; fila<9; fila++){ /* Se copia el sudoku sobre el tablero a evaluar */
        for (var columna=0; columna<9; columna++)
            if (sudoku[fila][columna] != 0) solucion[fila][columna] =
sudoku[fila][columna];
    }

    var numValido=true;
    do { /* Busca un número al azar para colocar en alguna celda */
        var posX = Math.floor(Math.random()*9); /* Una posición X de 0 a 8 */
        var posY = Math.floor(Math.random()*9); /* Una posición Y de 0 a 8 */
        var numero = Math.floor(Math.random()*9) + 1; /* Un número al azar de 1 a 9 */
        numValido=true; /* Chequea si el número no se repite ni vertical ni horizontalmente */
        for (var i=0; i<9; i++)
            if (solucion[posX][i] == numero || solucion[i][posY] == numero)
                numValido=false;
    }
}
```

```

        for (var cont=0; cont<9; cont++) if
(solucion[cont][posY]==numero || solucion[posX][cont]==numero)
numValido=false;
        if (numValido) solucion[posX][posY]=numero; /* Si el número no
se repite entonces lo coloca en el tablero */
    }while(!numValido);

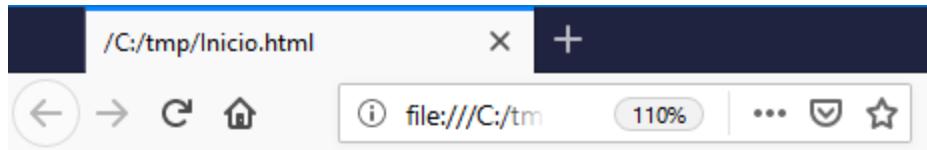
/* Chequea que NO se viole la regla de que cada uno de los 9 cuadros
internos no repita número */
for (var cuadroX=0; cuadroX<=6; cuadroX+=3)
    for (var cuadroY=0; cuadroY<=6; cuadroY+=3) {
        var numRepite=0;
        for (var valor=1; valor<=9; valor++) {
            numRepite=0;
            for (var posX=0; posX<3; posX++)
                for (var posY=0; posY<3; posY++) {
                    if
(solucion[cuadroX+posX][cuadroY+posY]==valor) numRepite++;
                    if (numRepite>1) break;
                }
            if (numRepite>1) /* Si detecta repetición, entonces
borra todos los números repetidos */
                for (var posX=0; posX<3; posX++)
                    for (var posY=0; posY<3; posY++)
                        if
(solucion[cuadroX+posX][cuadroY+posY]==valor)
solucion[cuadroX+posX][cuadroY+posY]=0;
            }
        }
finalizo=true; /* Chequea si se completó el sudoku completamente
*/
for (var posX=0; posX<9; posX++)
    for (var posY=0; posY<9; posY++)
        if (solucion[posX][posY]==0) finalizo=false;

ciclos++; /* Cada ciertos ciclos para destribar, borra la
tercera parte de lo completado */
if (ciclos%DESTRUYE==0)
    for (var posX=0; posX<9; posX++)
        for (var posY=0; posY<9; posY++)
            if (Math.floor(Math.random()*3)==0)
solucion[posX][posY]=0;
}while (!finalizo);

document.write("<p style='font-family:courier new;'>");
document.write("Ciclos totales: " + ciclos + "</p>");
for (var posX=0; posX<9; posX++){
    document.write("<p style='font-family:courier new;'>");
    for (var posY=0; posY<9; posY++)

```

```
        document.write(solucion[posX][posY] + " ") ;
        document.write("</p>") ;
    }
</script></head></html>
```



Ciclos totales: 14762

5 3 4 6 7 8 9 1 2

6 7 2 1 9 5 3 4 8

1 9 8 3 4 2 5 6 7

8 5 9 7 6 1 4 2 3

4 2 6 8 5 3 7 9 1

7 1 3 9 2 4 8 5 6

9 6 1 5 3 7 2 8 4

2 8 7 4 1 9 6 3 5

3 4 5 2 8 6 1 7 9

Ilustración 141: Sudoku resuelto

## Poner los barcos en el juego batalla naval

Directorio 036

```
<!DOCTYPE HTML><html><head><script>
/* Hacer un algoritmo que ponga los 5 barcos
   al azar en el tablero del juego de batalla naval */

//Crear el tablero del juego
var tableroJuego = inicializaTablero(10, 10);

//Poner cada barco: arreglo, tipo barco, número de huecos
poneBarco(tableroJuego, 'P', 5);
poneBarco(tableroJuego, 'G', 4);
poneBarco(tableroJuego, 'C', 3);
poneBarco(tableroJuego, 'S', 3);
poneBarco(tableroJuego, 'D', 2);
imprime(tableroJuego);

//Genera el tablero
function inicializaTablero(filas, columnas){
    var tablero = new Array();
    for (var fila=0; fila<filas; fila++){
        tablero[fila] = [];
        for (var columna=0; columna<columnas; columna++)
            tablero[fila][columna]='.';
    }
    return tablero;
}

function imprime(tablero){ //Imprime el tablero
    for (var fila=0; fila<tablero.length; fila++){
        document.write("<p style='font-family:courier new;'>");
        for (var col=0; col<tablero[0].length; col++)
            document.write(tablero[fila][col] + " ");
        document.write("</p>");
    }
    document.write("</p>");
}

//Pone el barco en una posición al azar
function poneBarco(tablero, simbolo, huecos){
    var orienta, fil, col, posValida;
    do{
        /* ¿Vertical u horizontal?
           0 a 0.4999 es vertical,
           0.5 a 1 es horizontal */
        orienta = Math.random();

        fil = Math.floor(Math.random()*tablero.length);
        col = Math.floor(Math.random()*tablero[0].length);
        ...
    } while (!posValida);
    ...
}
```

```

posValida = true; //La posición del barco es válida

//Chequea si no se sale del tablero
if (fil+huecos >= tablero.length && orienta<0.5)
    posValida=false; //La posición del barco es inválida
else if (col+huecos >= tablero[0].length && orienta>=0.5)
    posValida=false; //La posición del barco es inválida
else { //Chequea si ya ha sido ocupada esa parte
    if (orienta<0.5)
        for(var cont=0; cont<huecos; cont++) {
            if (tablero[fil+cont][col]!=='.')
                posValida=false;
        }
    else
        for(var cont=0; cont<huecos; cont++) {
            if (tablero[fil][col+cont]!=='.')
                posValida=false;
        }
}
}while(posValida==false);

if (orienta<0.5)
    for(var cont=0; cont<huecos; cont++)
        tablero[fil+cont][col]=simbolo;
else
    for(var cont=0; cont<huecos; cont++)
        tablero[fil][col+cont]=simbolo;
}

</script></head></html>

```

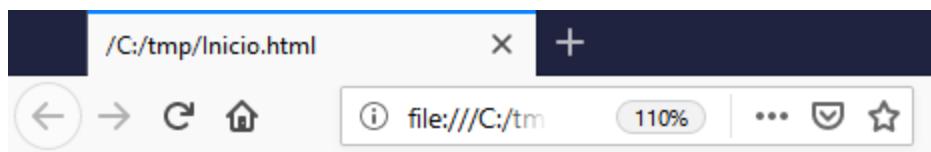


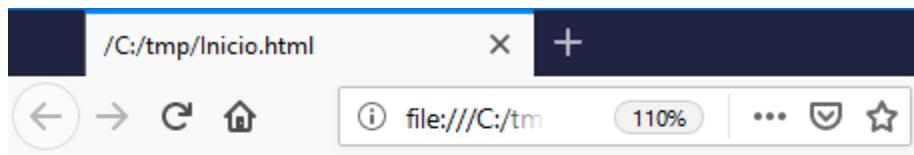
Ilustración 142: Barcos ubicados en el juego de mesa Batalla Naval

## Funciones que reciben distinto número de parámetros

Directorio 037

```
<!DOCTYPE HTML><html><head><script>
//Funciones que reciben distinto número de parámetros
UnaFuncion();
UnaFuncion("esto", "es", "una", "prueba");
UnaFuncion(1,2,3,4,5,6,7,8,9);
UnaFuncion("palabra", 5, "texto", 7, 8.56);

function UnaFuncion(){
    //total parámetros
    document.write("<br>total parámetros: " + arguments.length);
}
</script></head></html>
```



```
total parámetros: 0
total parámetros: 4
total parámetros: 9
total parámetros: 5
```

Ilustración 143: Número de parámetros variable en funciones.

Directorio 037

```
<!DOCTYPE HTML><html><head><script>
//Funciones que reciben distinto número de parámetros
FuncionDinamica("paso", "varios", "argumentos", 7, 1, "numeros", 3, "y
texto");

function FuncionDinamica(){
    for (var cont=0; cont < arguments.length; cont++)
        document.write(arguments[cont] + "<br>");
}
</script></head></html>
```

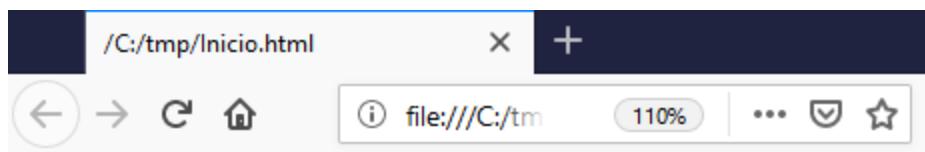


Ilustración 144: Muestra el contenido de cada parámetro de la función

## Funciones que reciben a su vez funciones por parámetros

Directorio 038

```
<!DOCTYPE HTML><html><head><script>
//Envia una función como parámetro a otra función

//Se envía una función que retorna el resultado de a - b
var prueba = Ejecuta( function(a,b){ return a-b; } )
document.write(prueba + "<br>");

//Se envía una función que retorna el resultado de a * b
prueba = Ejecuta( function(a,b){ return a*b; } )
document.write(prueba + "<br>");

//Se envía una función que retorna el resultado de a / b
prueba = Ejecuta( function(a,b){ return a/b; } )
document.write(prueba + "<br>");

//Esta función recibe como parámetro: ;una función!
function Ejecuta(unafuncion){
    var valorA = 15;
    var valorB = 10;
    //Dependiendo de la función recibida como parámetro, ejecuta la
    acción
    var resultado = unafuncion(valorA, valorB);
    return resultado;
}
</script></head></html>
```

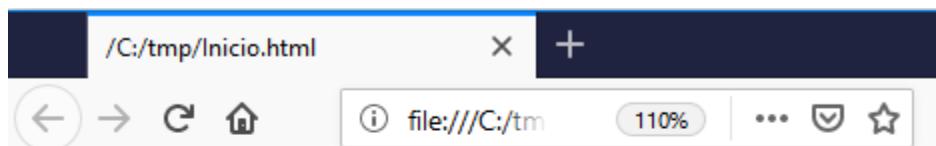


Ilustración 145: Envía una función como parámetro a otra función

```

<!DOCTYPE HTML><html><head><script>
//Envia una función como parámetro a otra función
var prueba = Ejecuta(5, 7, function(a,b){ return a-b; } )
document.write(prueba + "<br>");

prueba = Ejecuta( 10, 3, function(a,b){ return a*b; } )
document.write(prueba + "<br>");

prueba = Ejecuta( 9, 12, function(a,b){ return a/b; } )
document.write(prueba + "<br>");

//Esta función recibe como tercer parámetro: ;una función!
function Ejecuta(valorA, valorB, unafuncion){
    var resultado = unafuncion(valorA, valorB);
    return resultado;
}
</script></head></html>

```

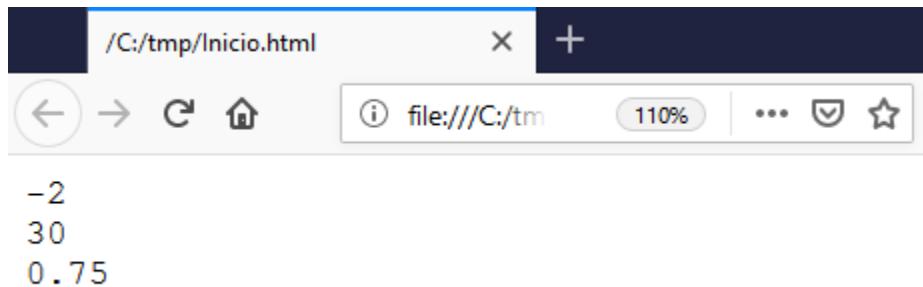


Ilustración 146: Envía una función como parámetro a otra función

## Operaciones de bit

### Convertir un entero a su representación binaria

Directorio 039

```
<!DOCTYPE HTML><html><head><script>
//Convertir un entero a su representación binaria
var valor = 17;
var resultado = valor.toString(2);
document.write(valor + " en binario es: " + resultado + "<br>");

//Conversión directa
var convierte = 890..toString(2);
document.write("En binario el número 890 es " + convierte + "<br>");
```

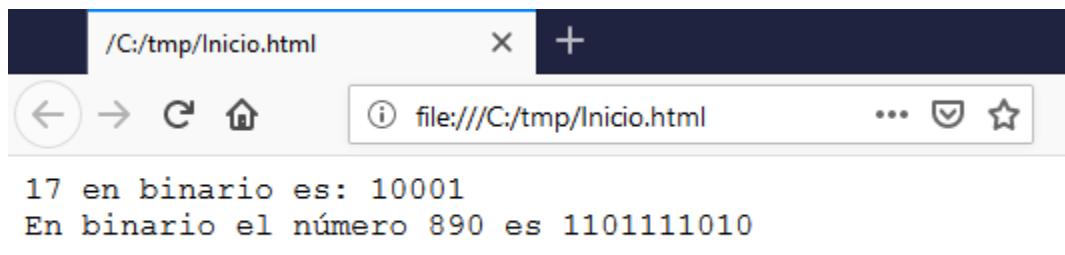


Ilustración 147: Convertir un entero a su representación binaria

## Operaciones OR, AND y XOR directas

Directorio 039

```
<!DOCTYPE HTML><html><head><script>
//Operaciones de bit
var valA = 25;
var valB = 20;
document.write(valA + " en binario es: " + valA.toString(2) + "<br>");
document.write(valB + " en binario es: " + valB.toString(2) + "<br>");

var valC = valA | valB; //Operación OR
document.write(valC + " en binario es: " + valC.toString(2) + "<br>");
```

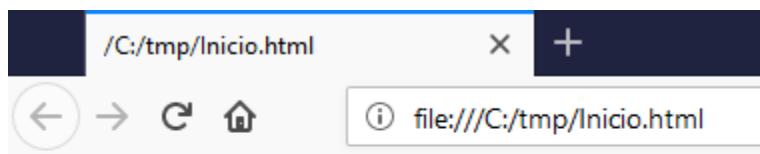


Ilustración 148: Operación OR

Directorio 039

```
<!DOCTYPE HTML><html><head><script>
//Operaciones de bit
var valA = 25;
var valB = 20;
document.write(valA + " en binario es: " + valA.toString(2) + "<br>");
document.write(valB + " en binario es: " + valB.toString(2) + "<br>");

var valC = valA & valB; //Operación AND
document.write(valC + " en binario es: " + valC.toString(2) + "<br>");
```

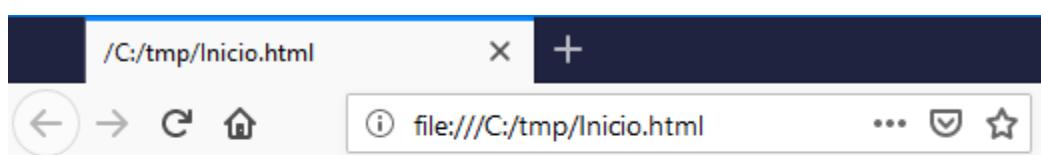


Ilustración 149: Operación AND

```
<!DOCTYPE HTML><html><head><script>
//Operaciones de bit
var valA = 25;
var valB = 20;
document.write(valA + " en binario es: " + valA.toString(2) + "<br>"); 
document.write(valB + " en binario es: " + valB.toString(2) + "<br>");

var valC = valA ^ valB; //Operación XOR
document.write(valC + " en binario es: " + valC.toString(2) + "<br>"); 
</script></head></html>
```

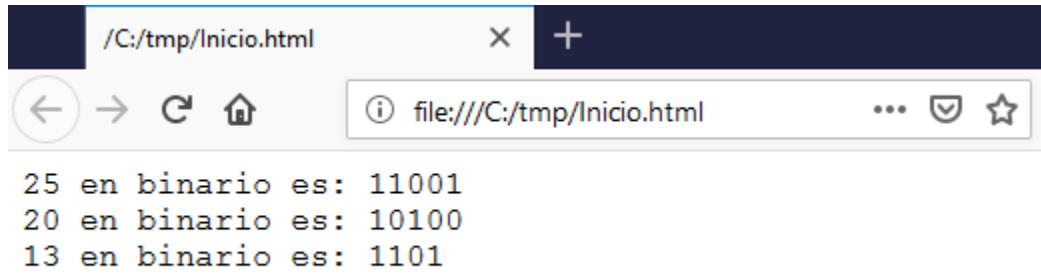


Ilustración 150: Operación XOR

## Multiplicar usando operaciones de bit

Directorio 039

```
<!DOCTYPE HTML><html><head><script>
//Operaciones de bit
var valA = 2;
document.write(valA + " en binario es: " + valA.toString(2) + "<br>");

var valB = valA << 1; //Multiplica por 2
document.write(valB + " en binario es: " + valB.toString(2) + "<br>");

var valC = valA << 2; //Multiplica por 4
document.write(valC + " en binario es: " + valC.toString(2) + "<br>");

var valD = valA << 3; //Multiplica por 8
document.write(valD + " en binario es: " + valD.toString(2) + "<br>");

</script></head></html>
```

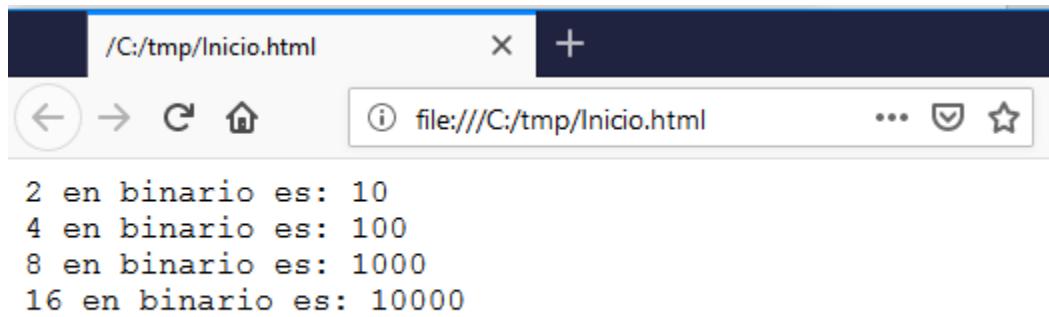


Ilustración 151: Multiplicación usando operaciones de bit

## Dividir usando operaciones de bit

Directorio 039

```
<!DOCTYPE HTML><html><head><script>
//Operaciones de bit
var valA = 64;
document.write(valA + " en binario es: " + valA.toString(2) + "<br>");

var valB = valA >> 1; //Divide entre 2
document.write(valB + " en binario es: " + valB.toString(2) + "<br>");

var valC = valA >> 2; //Divide entre 4
document.write(valC + " en binario es: " + valC.toString(2) + "<br>");

var valD = valA >> 3; //Divide entre 8
document.write(valD + " en binario es: " + valD.toString(2) + "<br>");

</script></head></html>
```

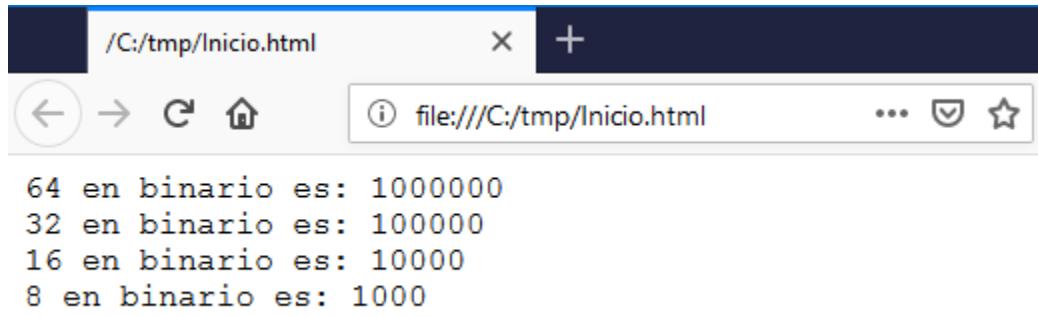


Ilustración 152: División usando operaciones de bit

## Intercambiar valores usando operaciones de bit

Directorio 039

```
<!DOCTYPE HTML><html><head><script>
var valA = 64;
var valB = 17;
document.write("valores: " + valA + " , " + valB + "<br>");

//Para intercambiar los valores de esas variables se puede hacer uso de
operadores de bit (XOR)
valA^=valB;
valB^=valA;
valA^=valB;
document.write("valores: " + valA + " , " + valB + "<br>");
</script></head></html>
```

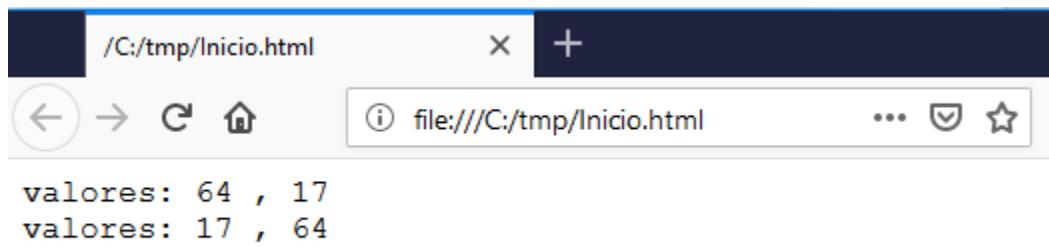


Ilustración 153: Intercambiar valores usando operaciones de bit

# Programación orientada a objetos

Se sigue la especificación ECMAScript 2015 (ES6)

## Definiendo clases, constructores y atributos

Directorio 040

```
<!DOCTYPE HTML><html><head><script>
//Definir una clase
class Gato {
    //Constructor de la clase
    constructor(nombre, sexo, fechanace, raza){
        this.nombre = nombre; //define el atributo y le da un valor
        this.sexo = sexo;
        this.fechanace = fechanace;
        this.raza = raza;
    }
}

//Instanciar un objeto de esa clase
var prueba = new Gato("Sally", "F", "10 de junio de 2010", "Criollo");

document.write(prueba.nombre + " , " + prueba.sexo + " , ");
document.write(prueba.fechanace + " , " + prueba.raza);
</script></head></html>
```

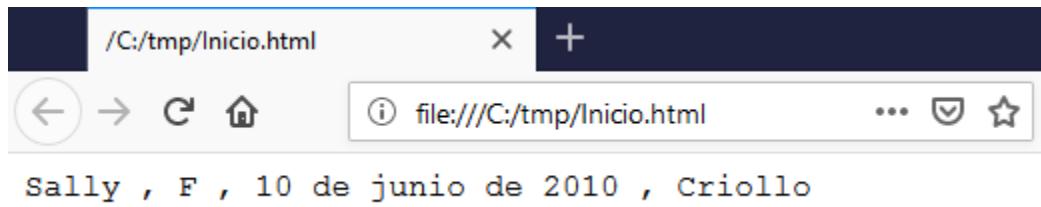


Ilustración 154: Valores de atributos

## Definiendo métodos

Directorio 040

```
<!DOCTYPE HTML><html><head><script>
//Definir una clase
class Geometria {
    //Define métodos
    AreaCuadrado(Lado) {
        return Lado*Lado;
    }

    //Define métodos
    AreaTriangulo(Base, Altura) {
        return Base*Altura/2;
    }
}

//Instanciar un objeto de esa clase
var probar = new Geometria();

document.write("Area del cuadrado es: " + probar.AreaCuadrado(9));
document.write("<br>Area del triangulo es: " +
probar.AreaTriangulo(3,4));
</script></head></html>
```

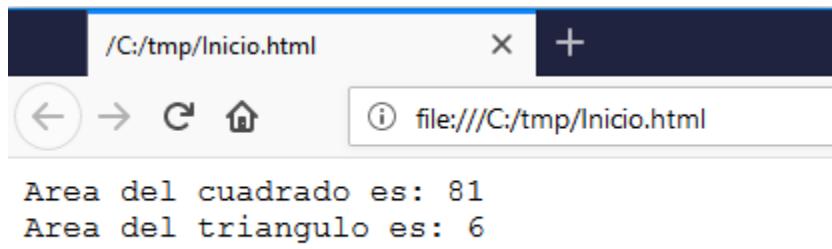


Ilustración 155: Métodos

## Métodos y atributos

Directorio 040

```
<!DOCTYPE HTML><html><head><script>
//Definir una clase
class Geometria {
    //Método que define el atributo Lado y le da un valor
    ValorLado(Lado) {
        this.Lado = Lado;
    }

    //Método que retorna el valor del área
    AreaCuadrado() {
        return this.Lado * this.Lado;
    }
}

//Instanciar un objeto de esa clase
var probar = new Geometria();
probar.ValorLado(11);
document.write("Área del Cuadrado es: " + probar.AreaCuadrado());
</script></head></html>
```

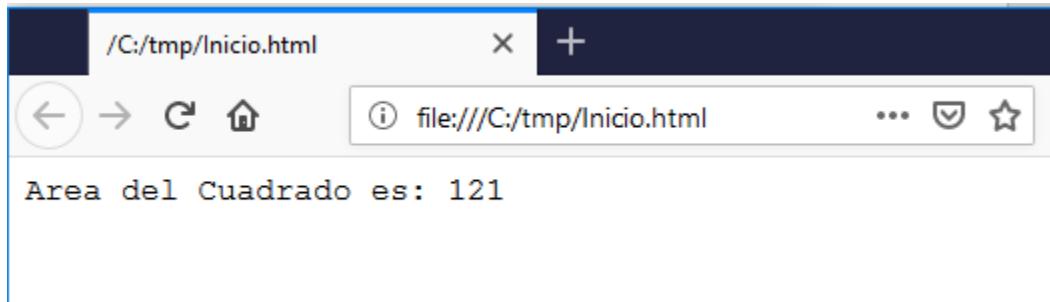


Ilustración 156: Métodos y atributos

## Atributos y métodos privados

En ES6 no se definen métodos ni atributos privados. Se hace uso de la convención \_

Directorio 040

```
<!DOCTYPE HTML><html><head><script>
//Definir una clase
class UnaClase {
    //La convención es poner un _ al principio
    //del nombre del método para indicar que es
    //un método privado
    _UnMetodo () {
        document.write("Método privado<br>");
    }

    _OtroMetodo () {
        document.write("Otro método privado<br>");
    }
}

//Instanciar un objeto de esa clase
var probar = new UnaClase();
probar._UnMetodo();
probar._OtroMetodo();

</script></head></html>
```

Es una convención, no es validado por el lenguaje. Luego el anterior programa ejecutará sin problemas.

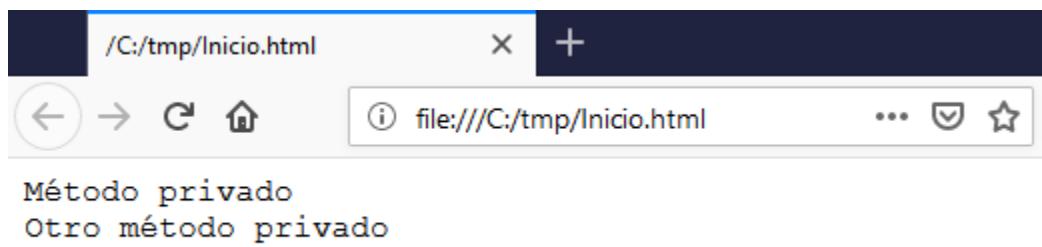


Ilustración 157: Métodos privados

## Getters y Setters

Directorio 040

```
<!DOCTYPE HTML><html><head><script>
class MiClase {
    constructor(texto) {
        this._texto = texto; //Define el atributo "privado"
    }

    //Getter
    get texto() {
        return this._texto;
    }

    //Setter
    set texto(valor) {
        this._texto = valor;
    }
}

probar = new MiClase('Grisú y Suini');
document.write(probar.texto + "<br>");
probar.texto = "Sally";
document.write(probar.texto + "<br>");
</script></head></html>
```

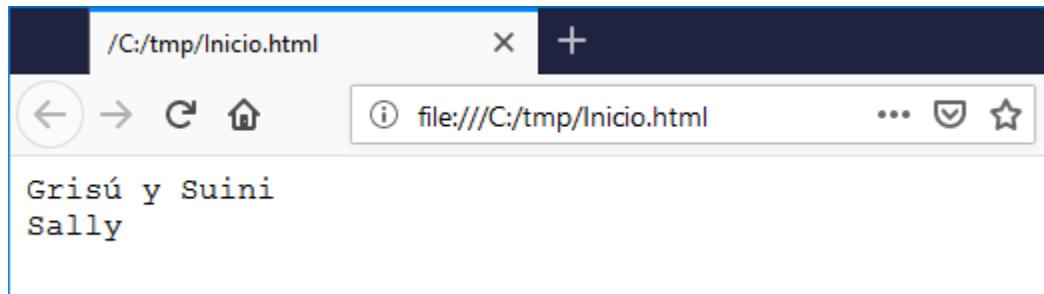


Ilustración 158: Getters y Setters

Podemos probar que los métodos get y set son llamados

Directorio 040

```
<!DOCTYPE HTML><html><head><script>
class MiClase {
    constructor(texto) {
        this._texto = texto; //Define el atributo "privado"
    }
    //Getter
    get texto() {
        document.write("Retorna valor <br>");
        return this._texto;
    }
    //Setter
    set texto(valor) {
        document.write("Cambia valor <br>");
        this._texto = valor;
    }
}

probar = new MiClase('Grisú y Suini');
document.write(probar.texto + "<br>");
probar.texto = "Sally";
document.write(probar.texto + "<br>");
</script></head></html>
```



Ilustración 159: Getters y Setters

# Herencia

Directorio 040

```
<!DOCTYPE HTML><html><head><script>
//Clase Madre
class Madre {
}

//Clase Hija (hereda de la madre)
class Hija extends Madre {
}

prueba = new Hija();

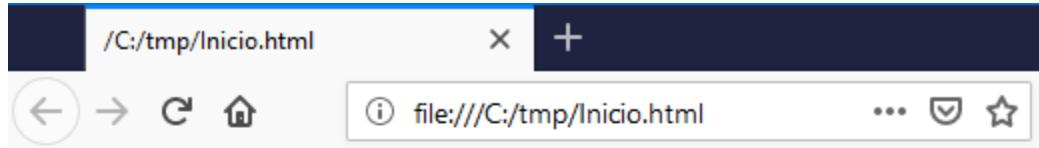
document.write("<br><br>prueba es una instancia de Hija: ");
document.write(prueba instanceof Hija); //Resultado true

document.write("<br><br>prueba es una instancia de Madre: ");
document.write(prueba instanceof Madre); //Resultado true

test = new Madre();

document.write("<br><br>test es una instancia de Hija: ");
document.write(test instanceof Hija); //Resultado false

document.write("<br><br>test es una instancia de Madre: ");
document.write(test instanceof Madre); //Resultado true
</script></head></html>
```



```
prueba es una instancia de Hija: true
prueba es una instancia de Madre: true
test es una instancia de Hija: false
test es una instancia de Madre: true
```

Ilustración 160: Herencia

## Herencia y constructor

Directorio 040

```
<!DOCTYPE HTML><html><head><script>
//Clase Madre
class Madre {
    constructor() {
        document.write("<br>Constructor clase madre");
    }
}

//Clase Hija (hereda de la madre)
class Hija extends Madre {
    constructor() {
        super(); //Llama al constructor de la clase madre
        document.write("<br>Constructor clase hija");
    }
}

prueba = new Hija(); //Imprime "Constructor clase madre" y "Constructor clase hija"
</script></head></html>
```



Ilustración 161: Herencia y constructor

Directorio 040

```
<!DOCTYPE HTML><html><head><script>
//Clase abuela
class Abuela {
    constructor() {
        document.write("<br>Constructor clase abuela");
    }
}

//Clase Madre
class Madre extends Abuela {
    constructor() {
        super();
        document.write("<br>Constructor clase madre");
    }
}
```

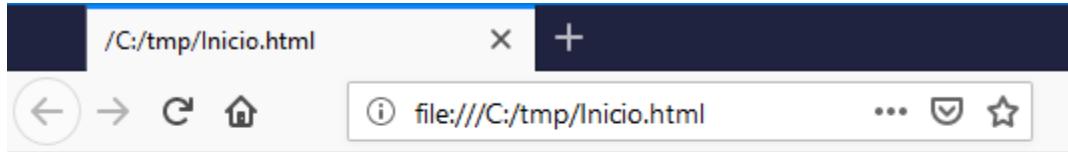
```

        }

//Clase Hija (hereda de la madre)
class Hija extends Madre {
    constructor() {
        super();
        document.write("<br>Constructor clase hija");
    }
}

prueba = new Hija();
/* Imprime:
   Constructor clase abuela
   Constructor clase madre
   Constructor clase hija
*/
</script></head></html>

```



Constructor clase abuela  
 Constructor clase madre  
 Constructor clase hija

Ilustración 162: Herencia y constructor

¡OJO! Debe hacer uso de la instrucción `super()` en el constructor de la clase hija (la que hereda) porque de lo contrario obtendrá un error y el script se detendrá:

`ReferenceError: must call super constructor before using |this| in Hija class constructor`

## Polimorfismo

No hay polimorfismo como tal en JavaScript como se puede ver en otros lenguajes de programación. El siguiente script pareciera que se implementa polimorfismo, pero NO funciona. Lo peor, es que no se genera mensaje de error en el navegador.

Directorio 040

```
<!DOCTYPE HTML><html><head><script>
/* El polimorfismo NO existe. Este script NO va a funcionar como se espera.
   Y no va a haber mensaje de error por parte del navegador */
class MiClase {
    MetodoA() {
        document.write("<br>Llama al método A con cero parámetros");
    }
    MetodoA(param1, param2) {
        document.write("<br>Llama al método A con dos parámetros");
    }
    MetodoA(param1) {
        document.write("<br>Llama al método A con un parámetro");
    }
    MetodoA(param1, param2, param3) {
        document.write("<br>Llama al método A con tres parámetros: " +
param1 + ", " + param2 + ", " + param3);
    }
}

prueba = new MiClase();
prueba.MetodoA();
prueba.MetodoA(2, 7);
prueba.MetodoA(8);
prueba.MetodoA(5, 9, 3);

/* El programa ejecutará así:
Llama al método A con tres parámetros: undefined, undefined, undefined
Llama al método A con tres parámetros: 2, 7, undefined
Llama al método A con tres parámetros: 8, undefined, undefined
Llama al método A con tres parámetros: 5, 9, 3 */
</script></head></html>
```

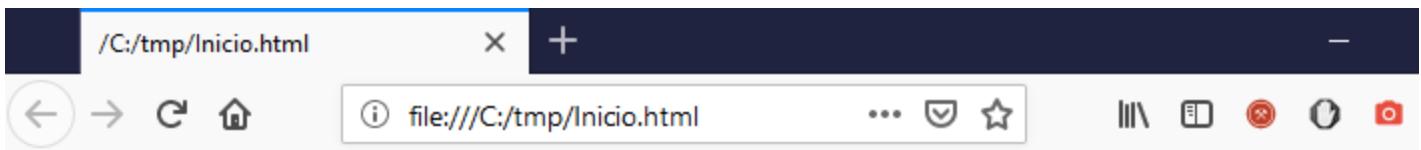


Ilustración 163: No hay polimorfismo en JavaScript

Nota: Existen técnicas para que JavaScript sea lo más parecido a un lenguaje orientado a objetos. El problema es que se requiere hacer uso de instrucciones anteriores a ES6. Al final se termina mezclando instrucciones anteriores con la nueva especificación, obteniendo una mezcla que puede ser difícil de entender.

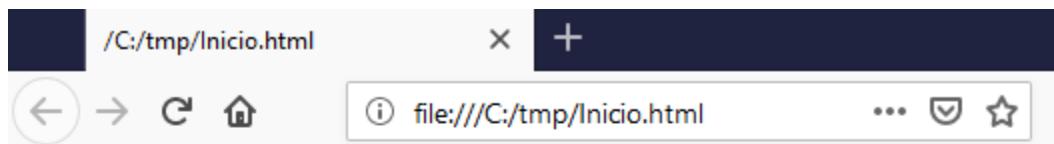
## Uso de typeof para detectar el tipo de dato de una variable

Directorio 041

```
<!DOCTYPE HTML><html><head><script>
var valorA;
document.write("<br>1. Tipo de valorA es " + typeof(valorA));
valorA = 157;
document.write("<br>2. Tipo de valorA es " + typeof(valorA));
valorA = "Había una vez...";
document.write("<br>3. Tipo de valorA es " + typeof(valorA));
valorA = 'K';
document.write("<br>4. Tipo de valorA es " + typeof(valorA));
valorA = true;
document.write("<br>5. Tipo de valorA es " + typeof(valorA));
valorA = new Date();
document.write("<br>6. Tipo de valorA es " + typeof(valorA));
valorA = 5/0; //Infinito
document.write("<br>7. Tipo de valorA es " + typeof(valorA));
valorA = Math.asin(39); //NaN
document.write("<br>8. Tipo de valorA es " + typeof(valorA));

/* Resultado:

1. Tipo de valorA es undefined
2. Tipo de valorA es number
3. Tipo de valorA es string
4. Tipo de valorA es string
5. Tipo de valorA es boolean
6. Tipo de valorA es object
7. Tipo de valorA es number
8. Tipo de valorA es number */
</script></head></html>
```



```
1. Tipo de valorA es undefined
2. Tipo de valorA es number
3. Tipo de valorA es string
4. Tipo de valorA es string
5. Tipo de valorA es boolean
6. Tipo de valorA es object
7. Tipo de valorA es number
8. Tipo de valorA es number
```

Ilustración 164: typeof

```
<!DOCTYPE HTML><html><head><script>
class MiClase {
}

prueba = new MiClase();

//Imprime: prueba es de tipo: object
document.write("prueba es de tipo: " + typeof(prueba));
</script></head></html>
```

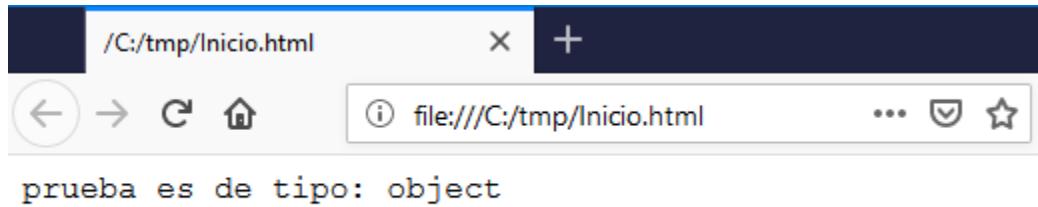


Ilustración 165: typeof de objetos

## Árbol binario

Con tres nodos se puede crear un árbol binario: raíz, rama izquierda, rama derecha

Directorio 042

```
<!DOCTYPE HTML><html><head><script>
//Definir una clase
class Nodo {
    //Constructor de la clase
    constructor(valor, izq, der) {
        this.valor = valor;
        this.izquierdo = izq;
        this.derecho = der;
    }
}

var NodoRaiz = new Nodo(1, null, null); //Nodo raíz
var NodoIzq = new Nodo(2, null, null); //Rama izquierda
var NodoDer = new Nodo(3, null, null); //Rama derecha

//El nodo raíz conecta a la izquierda y derecha
NodoRaiz.izquierdo = NodoIzq;
NodoRaiz.derecho = NodoDer;

//Imprime los valores del árbol
document.write(NodoRaiz.valor + "<br>");
document.write(NodoRaiz.izquierdo.valor + "<br>");
document.write(NodoRaiz.derecho.valor + "<br>");
</script></head></html>
```

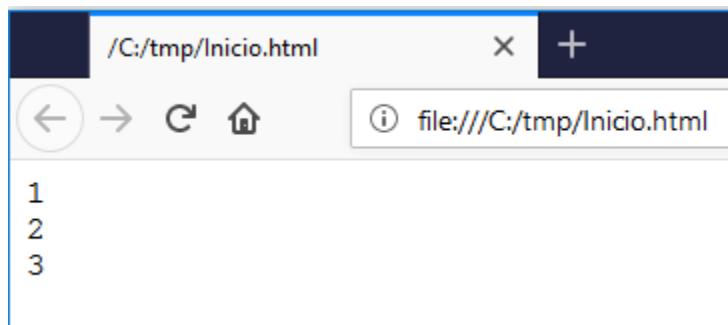


Ilustración 166: Recorrido de árbol binario

## Recorrido en pre-orden

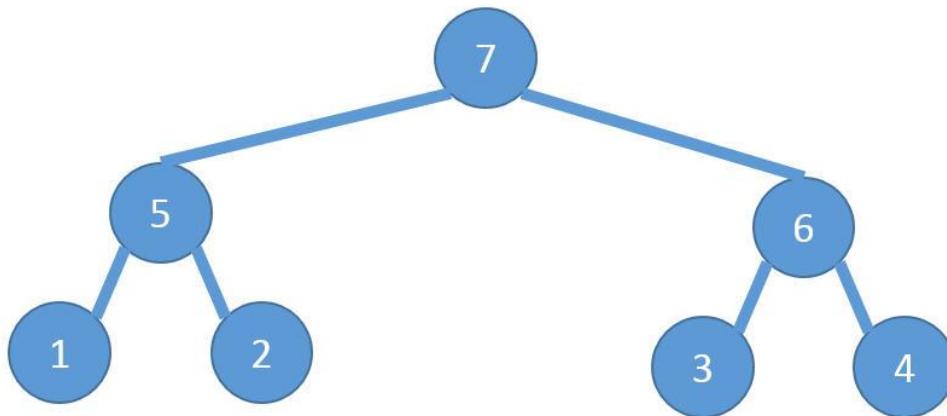


Ilustración 167: Árbol binario

Directorio 042

```
<!DOCTYPE HTML><html><head><script>
//Definir una clase
class Nodo {
    //Constructor de la clase
    constructor(valor, izq, der) {
        this.valor = valor;
        this.izquierdo = izq;
        this.derecho = der;
    }
}

//Se genera el árbol binario
var Nodo1 = new Nodo(1, null, null);
var Nodo2 = new Nodo(2, null, null);
var Nodo3 = new Nodo(3, null, null);
var Nodo4 = new Nodo(4, null, null);
var Nodo5 = new Nodo(5, Nodo1, Nodo2);
var Nodo6 = new Nodo(6, Nodo3, Nodo4);
var Nodo7 = new Nodo(7, Nodo5, Nodo6);

//Lo recorre en preorden
preorden(Nodo7); //7,5,1,2,6,3,4

function preorden(nodo) {
    if (nodo==null) return;
    document.write(" " + nodo.valor);
    preorden(nodo.izquierdo);
    preorden(nodo.derecho);
}
</script></head></html>
```

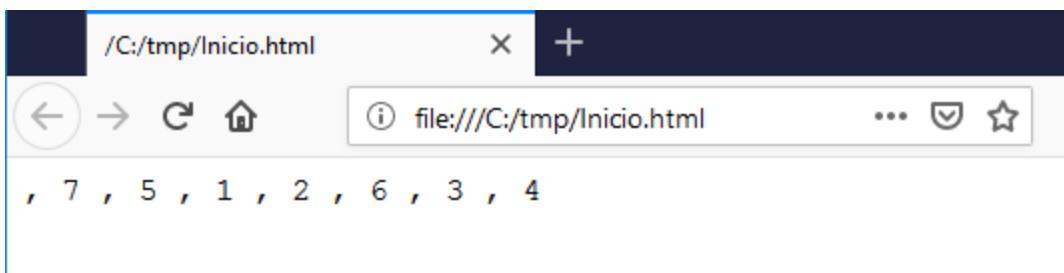


Ilustración 168: Recorrido en pre-orden

## Recorrido en in-orden

Directorio 042

```
<!DOCTYPE HTML><html><head><script>
//Definir una clase
class Nodo {
    //Constructor de la clase
    constructor(valor, izq, der) {
        this.valor = valor;
        this.izquierdo = izq;
        this.derecho = der;
    }
}

//Se genera el árbol binario
var Nodo1 = new Nodo(1, null, null);
var Nodo2 = new Nodo(2, null, null);
var Nodo3 = new Nodo(3, null, null);
var Nodo4 = new Nodo(4, null, null);
var Nodo5 = new Nodo(5, Nodo1, Nodo2);
var Nodo6 = new Nodo(6, Nodo3, Nodo4);
var Nodo7 = new Nodo(7, Nodo5, Nodo6);

inorden(Nodo7); //1,5,2,7,3,6,4

function inorden(nodo){
    if (nodo==null) return;
    inorden(nodo.izquierdo);
    document.write(" , " + nodo.valor);
    inorden(nodo.derecho);
}
</script></head></html>
```

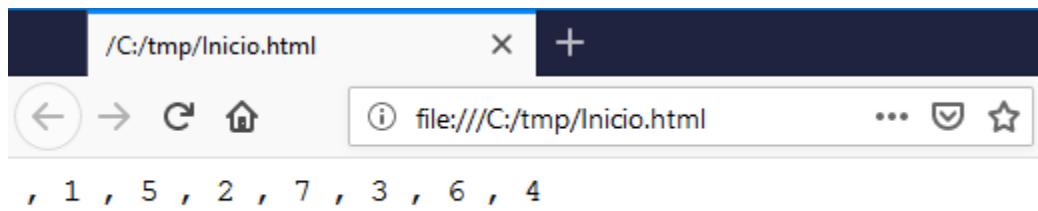


Ilustración 169: Recorrido en in-orden

## Recorrido en post-orden

Directorio 042

```
<!DOCTYPE HTML><html><head><script>
//Definir una clase
class Nodo {
    //Constructor de la clase
    constructor(valor, izq, der) {
        this.valor = valor;
        this.izquierdo = izq;
        this.derecho = der;
    }
}

//Se genera el árbol binario
var Nodo1 = new Nodo(1, null, null);
var Nodo2 = new Nodo(2, null, null);
var Nodo3 = new Nodo(3, null, null);
var Nodo4 = new Nodo(4, null, null);
var Nodo5 = new Nodo(5, Nodo1, Nodo2);
var Nodo6 = new Nodo(6, Nodo3, Nodo4);
var Nodo7 = new Nodo(7, Nodo5, Nodo6);

postorden(Nodo7); //1,2,5,3,4,6,7

function postorden(nodo){
    if (nodo==null) return;
    postorden(nodo.izquierdo);
    postorden(nodo.derecho);
    document.write(" , " + nodo.valor);
}
</script></head></html>
```

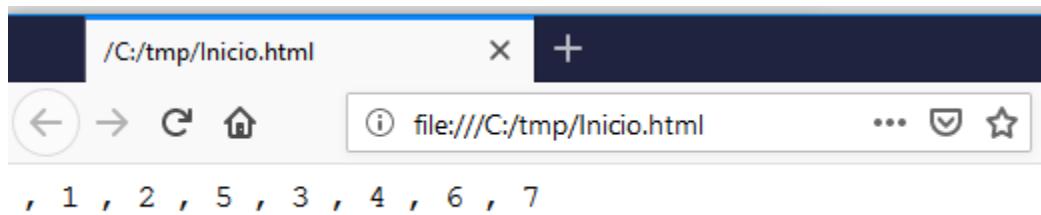


Ilustración 170: Recorrido en post-orden

## Ordenación usando un árbol binario

Directorio 042

```
<!DOCTYPE HTML><html><head><script> //Ordenación usando árbol binario
class Nodo {
    constructor(valor, izq, der) {
        this.valor = valor;
        this.izquierdo = izq;
        this.derecho = der;
    }
}

//Un arreglo con valores no ordenados
var arreglo = [ 15, 21, 17, 32, 48, 13, 29, 44, 19, 16, 3, -1, 9, 0];

//Genera el árbol binario
var NodoRaiz = new Nodo(arreglo[0], null, null);
for(var cont=1; cont<arreglo.length; cont++){
    var pasea = NodoRaiz;
    while (pasea!==null){
        if (arreglo[cont] < pasea.valor){ //Pone valor en la rama
izquierda
            if (pasea.izquierdo!==null)
                pasea = pasea.izquierdo;
            else{ //Crea nodo con el nuevo valor
                pasea.izquierdo = new Nodo(arreglo[cont], null, null);
                break;
            }
        }
        else{ //Pone valor en la rama derecha
            if (pasea.derecho!==null)
                pasea = pasea.derecho;
            else{ //Crea nodo con el nuevo valor
                pasea.derecho = new Nodo(arreglo[cont], null, null);
                break;
            }
        }
    } //Cierra el while que navega por el árbol binario
} //Cierra el for que pasea por todo el arreglo unidimensional no ordenado

//Recorre en in-orden el árbol binario y los datos están ordenados
inorden(NodoRaiz);

function inorden(nodo){
    if (nodo==null) return;
    inorden(nodo.izquierdo);
    document.write(", " + nodo.valor);
    inorden(nodo.derecho);
}
</script></head></html>
```



Ilustración 171: Ordenación usando un árbol binario

## Lista de objetos. Pilas.

Directorio 043

```
<!DOCTYPE HTML><html><head><script>
class Nodo {
    constructor(valorA, valorB) {
        this.valA = valorA;
        this.valB = valorB;
    }
}

var lista = new Array(); //Un arreglo, trabaja como lista de objetos
lista.push(new Nodo(17, "a"));
lista.push(new Nodo(43, "z"));
lista.push(new Nodo(85, "d"));

//Recorre la lista
for (cont=0; cont<lista.length; cont++)
    document.write(lista[cont].valA + ", " + lista[cont].valB + "<br>");
</script></head></html>
```

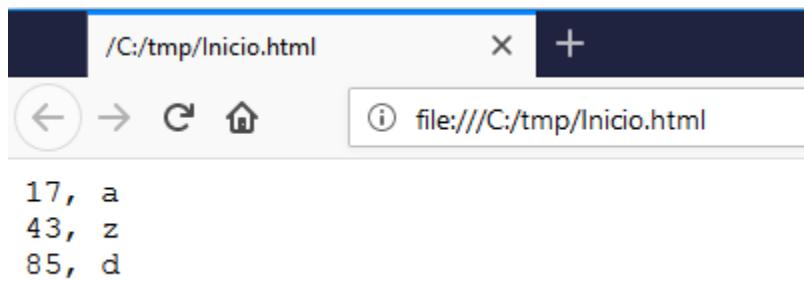


Ilustración 172: Lista de objetos. Pilas.

Directorio 043

```
<!DOCTYPE HTML><html><head><script>
class Nodo {
    constructor(valorA, valorB) {
        this.valA = valorA;
        this.valB = valorB;
    }
}

//Trabaja como una pila
var lista = new Array();
lista.push(new Nodo(17, "a"));
lista.push(new Nodo(43, "z"));
lista.push(new Nodo(85, "d"));

lista.pop(); //quita el último elemento adicionado
```

```
for (cont=0; cont<lista.length; cont++)  
    document.write(lista[cont].valA + " " + lista[cont].valB + "<br>");  
</script></head></html>
```

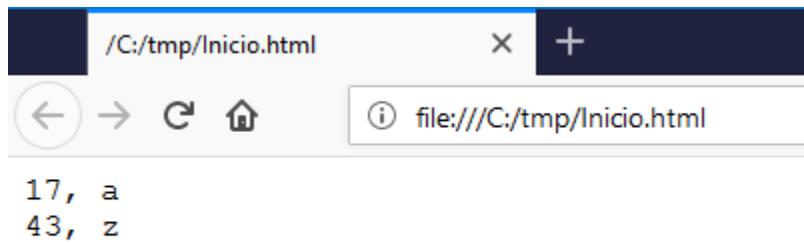


Ilustración 173: Lista de objetos. Pilas.

## Lista de objetos.

Directorio 044

```
<!DOCTYPE HTML><html><head><script>
class Nodo {
    constructor(valorA, valorB) {
        this.valA = valorA;
        this.valB = valorB;
    }
}

var lista = new Array(); //Un arreglo, trabaja como lista de objetos
lista.push(new Nodo(17, "a"));
lista.push(new Nodo(43, "z"));
lista.push(new Nodo(85, "d"));

lista.unshift(new Nodo(92, "k")); //Agrega un nodo al inicio

for (cont=0; cont<lista.length; cont++)
    document.write(lista[cont].valA + ", " + lista[cont].valB + "<br>");
</script></head></html>
```

Directorio 044

```
<!DOCTYPE HTML><html><head><script>
class Nodo {
    constructor(valorA, valorB) {
        this.valA = valorA;
        this.valB = valorB;
    }
}

var lista = new Array(); //Un arreglo, trabaja como lista de objetos
lista.push(new Nodo(17, "a"));
lista.push(new Nodo(43, "z"));
lista.push(new Nodo(85, "d"));

lista.shift(); //quita el primer nodo de la lista

for (cont=0; cont<lista.length; cont++)
    document.write(lista[cont].valA + ", " + lista[cont].valB + "<br>");
</script></head></html>
```

## Instrucciones que ejecutan controlando el tiempo

Directorio 045

```
<!DOCTYPE HTML><html><head><script>
document.write("Imprimirá en 3 segundos:");
setTimeout(Imprime, 3000); //3000ms = 3segundos

//Esta función ejecutará en 3 segundos
function Imprime(){
    document.body.innerHTML += ("<br>Esta es una prueba");
}
</script></head></html>
```

Directorio 045

```
<!DOCTYPE HTML><html><head><script>
//Cada 100 milisegundos se llamará a la función Imprime()
setInterval(Imprime, 100);

function Imprime(){
    document.body.innerHTML += ("<br>Esta es una prueba");
}
</script></head></html>
```

Directorio 045

```
<!DOCTYPE HTML><html><head><script>
var contador = 1;

//Cada 500 milisegundos se llamará a la función setInterval
var ejecuta = setInterval(Imprime, 500);

function Imprime(){
    document.body.innerHTML += ("<br>" + contador);
    contador++;

    //La función deja de llamarse
    if (contador == 10) clearInterval(ejecuta);
}
</script></head></html>
```

```
<!DOCTYPE HTML><html><head></head><body>
<div id="caja"></div>
<script>
/* Trata de mostrar los primos desde el 10.000 hasta un millón.
Eso toma bastante tiempo por lo que la ventana del navegador
se congela, no muestra nada y saldrá un aviso de si decide
detener el script. Al detener se muestran los números primos
que alcanzó a calcular */
var numero = 10000;
while(numero<1000000){
    if (EsPrimo(numero)) caja.innerHTML = caja.innerHTML + numero +
"<br>";
    numero++;
}

function EsPrimo(num){ //Retorna true si el número enviado por parámetro
es primo
    if (num<=1) return false;
    if (num==2) return true;
    if (num%2==0) return false;
    for (var divide = 3; divide <= Math.sqrt(num); divide+=2)
        if (num%divide == 0) return false;
    return true;
}
</script></body></html>
```

```
<!DOCTYPE HTML><html><head></head><body>
<div id="caja"></div>
<script>
/* Muestra los primos desde el 10.000 hasta un millón.
Eso toma bastante tiempo pero al usar setInterval se
pueden mostrar poco a poco los números sin congelar
la pestaña del navegador. */
var numero = 10000;
var ejecuta = setInterval(Imprime, 100);
var caja = document.getElementById("caja");

function Imprime(){
    if (EsPrimo(numero)) caja.innerHTML = caja.innerHTML + numero +
"<br>";
    numero++;
}

function EsPrimo(num){ //Retorna true si el número enviado por parámetro
es primo
    if (num<=1) return false;
    if (num==2) return true;
    if (num%2==0) return false;
    for (var divide = 3; divide <= Math.sqrt(num); divide+=2)
        if (num%divide == 0) return false;
    return true;
}
</script></body></html>
```

## Uso de JSON

Directorio 046

```
<!DOCTYPE HTML><html><body><script>
//Primer ejemplo de uso de JSON
var dato = { "id":13, "nombre":"Rafael Alberto Moreno", "tiposangre":"O+", 
"altura":1.80 };
document.write(dato.nombre + "<br>"); 
document.write(dato.tiposangre + "<br>"); 
document.write(dato.id + "<br>"); 
document.write(dato.altura + "<br>");

/* Imprime
Rafael Alberto Moreno
O+
13
1.8
*/
</script></body></html>
```

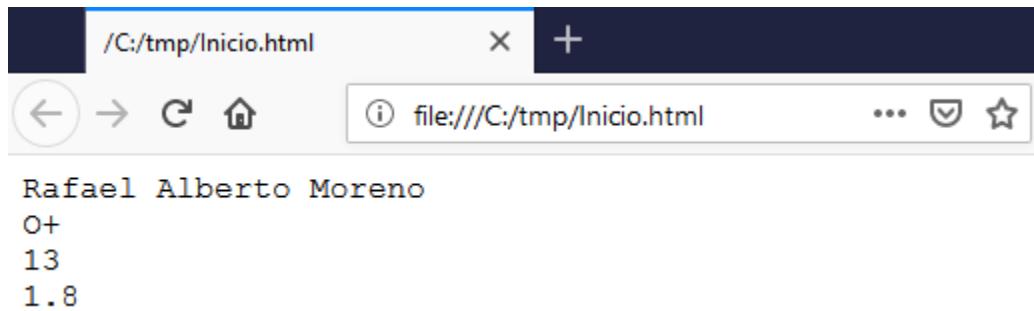


Ilustración 174: Primer ejemplo de uso de JSON

Directorio 046

```
<!DOCTYPE HTML><html><body><script>
//Segundo ejemplo de uso de JSON
var dato = { "mensaje": function(){ document.write("Hola Mundo"); } }
dato.mensaje();

/* Imprime:
   Hola Mundo
*/
</script></body></html>
```

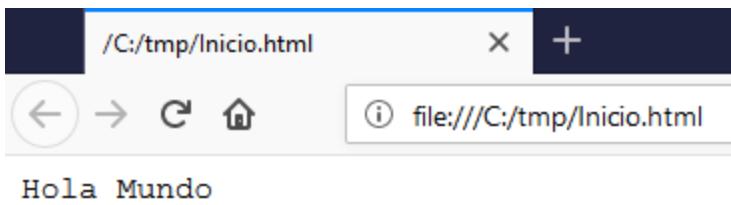


Ilustración 175: Segundo ejemplo de uso de JSON

Directorio 046

```
<!DOCTYPE HTML><html><body><script>
//Tercer ejemplo de uso de JSON
var dato = {"nombre":"Rafael", "mensaje": function(){
document.write(dato.nombre); } }
dato.mensaje();

/* Imprime:
   Rafael
*/
</script></body></html>
```

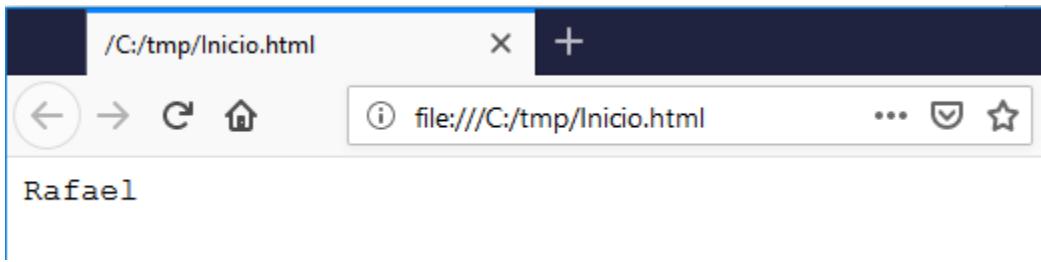


Ilustración 176: Tercer ejemplo de uso de JSON

Directorio 046

```
<!DOCTYPE HTML><html><body><script>
//Cuarto ejemplo de uso de JSON
var animal =
{ "Aves": [
    {"nombre":"Cuervo", "esperanzavida":70 },
    {"nombre":"Loro Gris", "esperanzavida":80 },
    {"nombre":"Zopilote de Turquía", "esperanzavida":119 } ]
};

document.write(animal.Aves[2].esperanzavida);

/* Imprime:
   119
*/
</script></body></html>
```

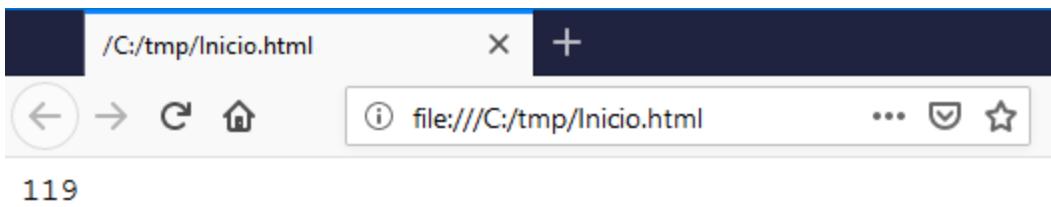


Ilustración 177: Cuarto ejemplo de uso de JSON

Directorio 046

```
<!DOCTYPE HTML><html><body><script>
//Quinto ejemplo de uso de JSON
var animal =
{ "Aves": [
    {"nombre":"Cuervo", "esperanzavida":70 },
    {"nombre":"Loro Gris", "esperanzavida":80 },
    {"nombre":"Zopilote de Turquí", "esperanzavida":118 } ],
  "Reptiles": [
    {"nombre":"Tortuga radiada", "esperanzavida":100 },
    {"nombre":"Cocodrilo", "esperanzavida":50 },
    {"nombre":"Dragón de Komodo", "esperanzavida":30 } ]
};
document.write(animal.Aves[0].nombre + "<br>");
document.write(animal.Reptiles[1].nombre);

/* Imprime:
   Cuervo
   Cocodrilo
*/
</script></body></html>
```

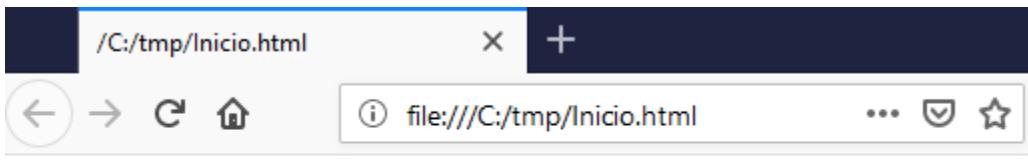


Ilustración 178: Quinto ejemplo de uso de JSON

Directorio 046

```
<!DOCTYPE HTML><html><body><script>
//Sexto ejemplo de uso de JSON
var animal =
{ "Aves": [
    {"nombre":"Cuervo", "esperanzavida":70 },
```

```

        {"nombre":"Loro Gris", "esperanzavida":80 },
        {"nombre":"Zopilote de Turquía", "esperanzavida":118 } ],
    "Reptiles": [
        {"nombre":"Tortuga radiada", "esperanzavida":100 },
        {"nombre":"Cocodrilo", "esperanzavida":50 },
        {"nombre":"Dragón de Komodo", "esperanzavida":30 } ]
};

for (var cont=0; cont < animal.Aves.length; cont++) {
    document.write(animal.Aves[cont].nombre + " vive hasta " );
    document.write(animal.Aves[cont].esperanzavida + " años <br>" );
}

/* Imprime:
Cuervo vive hasta 70 años
Loro Gris vive hasta 80 años
Zopilote de Turquía vive hasta 118 años
*/
</script></body></html>

```

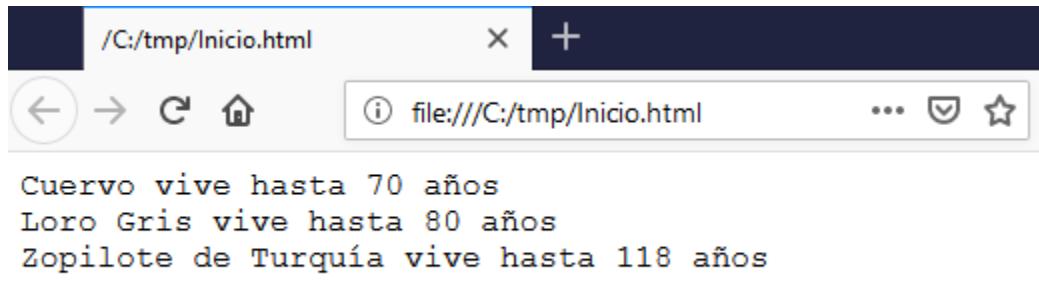


Ilustración 179: Sexto ejemplo de uso de JSON

Directorio 046

```

<!DOCTYPE HTML><html><body><script>
//Séptimo ejemplo de uso de JSON
var clima = [
    {"temperaturas": [23, 27, 29, 21] },
    {"temperaturas": [14, 18, 15, 11] },
    {"temperaturas": [33, 32, 35, 30] } ];

//Va de grupo en grupo de temperaturas
for (var cont=0; cont < clima.length; cont++) {

    //va de ítem en ítem de cada grupo de temperaturas
    for (var pos=0; pos < clima[cont].temperaturas.length; pos++)
        document.write(clima[cont].temperaturas[pos] + ", ");
    document.write("<br>");
}

```

```

/* Imprime:
23, 27, 29, 21,
14, 18, 15, 11,
33, 32, 35, 30,
*/
</script></body></html>

```

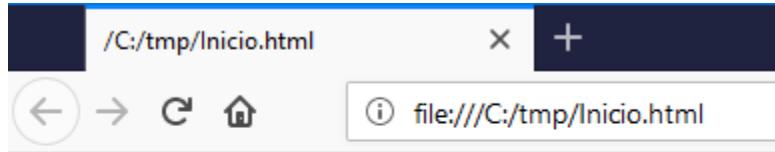


Ilustración 180: Séptimo ejemplo de uso de JSON

Directorio 046

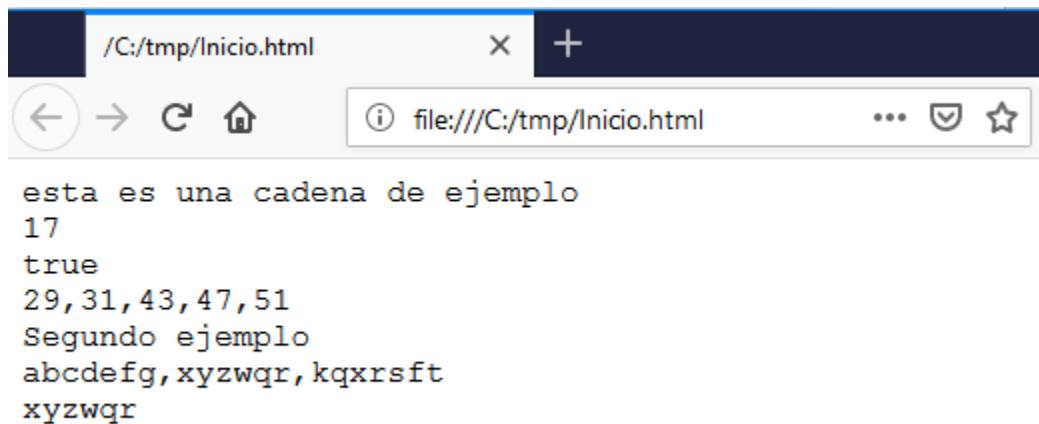
```

<!DOCTYPE HTML><html><head></head><body><script>
//Octavo ejemplo de uso de JSON
var Datos =
{
    "cadenaA": "esta es una cadena de ejemplo",
    "valorA": 17,
    "valorB": true,
    "valorC": [ 29, 31, 43, 47, 51],
    "otros": {
        "cadenaB": "Segundo ejemplo",
        "valores": [ "abcdefg", "xyzwqr", "kqxrsft" ]
    }
};

document.write(Datos.cadenaA + "<br>");
document.write(Datos.valorA + "<br>");
document.write(Datos.valorB + "<br>");
document.write(Datos.valorC + "<br>");
document.write(Datos.otros.cadenaB + "<br>");
document.write(Datos.otros.valores + "<br>");
document.write(Datos.otros.valores[1] + "<br>");

/* Imprime:
esta es una cadena de ejemplo
17
true
29,31,43,47,51
Segundo ejemplo
abcdefg,xyzwqr,kqxrsft
xyzwqr
*/
</script></body></html>

```



A screenshot of a web browser window titled "/C:/tmp/Inicio.html". The address bar shows "file:///C:/tmp/Inicio.html". The page content displays the following JSON output:

```
esta es una cadena de ejemplo
17
true
29,31,43,47,51
Segundo ejemplo
abcdefg,xyzwqr,kqxrsft
xyzwqr
```

Ilustración 181: Octavo ejemplo de uso de JSON

## Control sobre la página HTML

JavaScript permite un control completo sobre los objetos que hay en la página HTML.

Al dar clic en el botón muestra una ventana emergente de alerta. Se programa el evento “onClick”

Directorio 047

```
<!DOCTYPE HTML><html><head></head><body>
<input type="button" id="boton" value="Tocar" onClick="alert('Hola
mundo');"
</body></html>
```

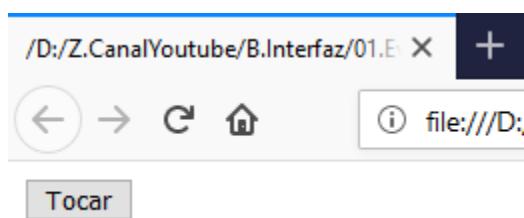


Ilustración 182: Botón

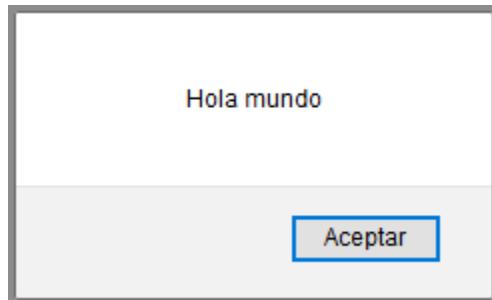


Ilustración 183: Ventana emergente generada por JavaScript

El siguiente código hace lo mismo haciendo uso de funciones

Directorio 047

```
<!DOCTYPE HTML><html><head><script>
//Al dar clic en el botón se ejecuta esta función
function aviso(){
    alert("Hola mundo");
}
</script></head><body>
<input type="button" id="boton" value="Tocar2" onClick="aviso()" />
</body></html>
```

Funciona también para enlaces

```
<!DOCTYPE HTML><html><head></head><body>
<a href="javascript:alert('Hola mundo')">De clic aquí</a>
</body></html>
```

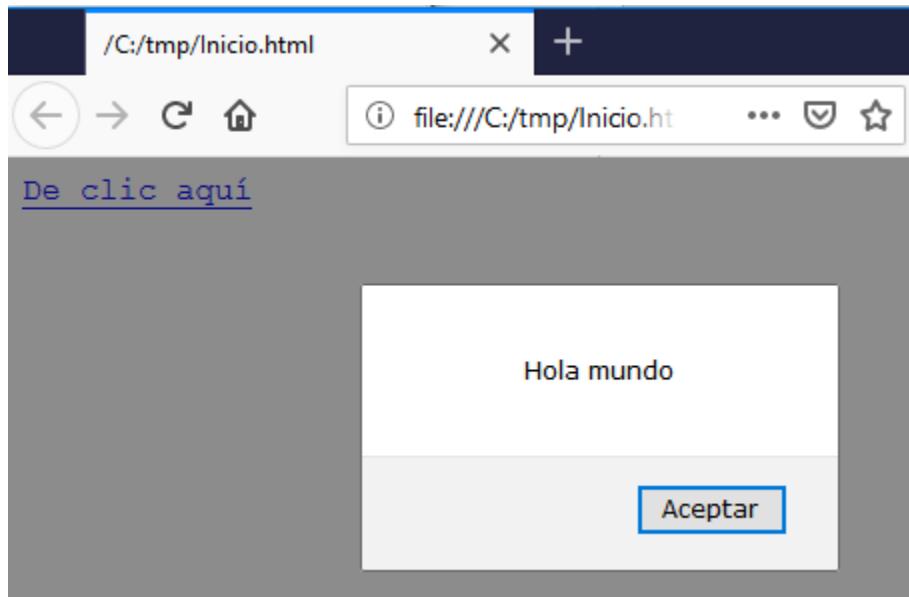


Ilustración 184: Ventana emergente al dar clic en un hipervínculo

Y se puede separar el código HTML de JavaScript. Primero el código JavaScript en su propio archivo (para este ejemplo es saludo.js)

```
/* Funciones JavaScript en un archivo externo con extensión .js */
function imprimir(){
    alert("Esto es una prueba");
}
```

Y el código HTML que lo llama

```
<!DOCTYPE HTML><html><head><script src="saludo.js"></script></head><body>
<a href="javascript:imprimir()">De clic aquí</a>
</body></html>
```

## Captura el evento de dar clic en un <div>

Puede activarse cuando el usuario de clic en el interior de <div>

Directorio 047

```
<!DOCTYPE HTML><html><head><style type="text/css">
    div{
        width: 230px;
        height: 230px;
        margin: 20px;
        background-color: lightgreen;
    }
</style>
<script>
//Al dar clic en el <div> se ejecuta esta función
function clicCaja(){
    alert("Dio clic dentro de un <div>");
}
</script></head><body>
    <div id="caja1" onclick="clicCaja()"></div>
    <div id="caja2" onclick="clicCaja()"></div>
    <div id="caja3" onclick="clicCaja()"></div>
    <div id="caja4" onclick="clicCaja()"></div>
</body></html>
```

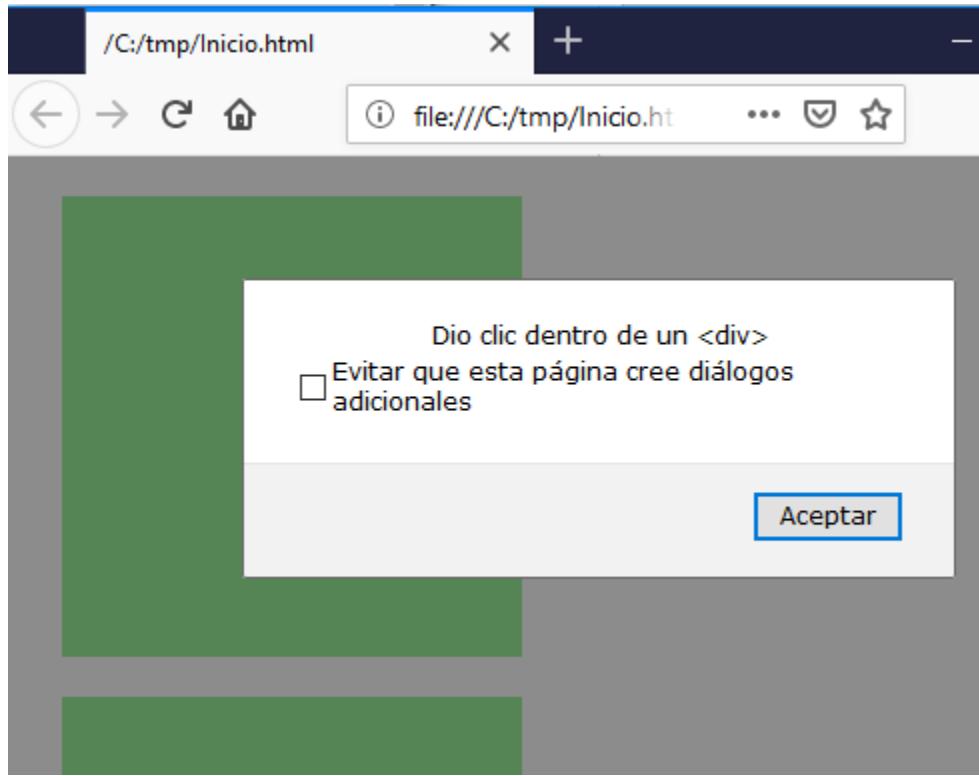


Ilustración 185: Ventana emergente al dar clic en un <div>

Puede saberse en qué <div> el usuario da clic. Cada <div> debe tener su propio id

Directorio 047

```
<!DOCTYPE HTML><html><head><style type="text/css">
    div{
        width: 70px;
        height: 70px;
        margin: 20px;
        background-color: lightgreen;
    }
</style>
<script>
    function divclic(caja){
        alert("<div> tocado es: " + caja.id);
    }
</script></head><body>
    <div id="caja1" onclick="divclic(this)"></div>
    <div id="caja2" onclick="divclic(this)"></div>
    <div id="caja3" onclick="divclic(this)"></div>
    <div id="caja4" onclick="divclic(this)"></div>
</body></html>
```

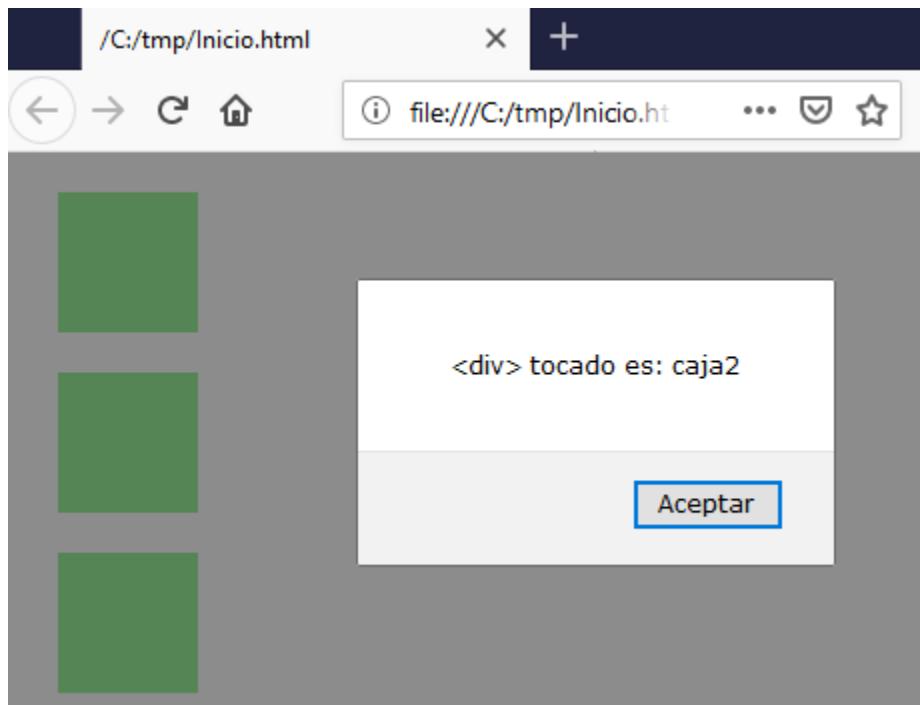


Ilustración 186: Ventana emergente al dar clic en un <div>

## Captura el evento de dar doble clic en un <div>

Directorio 047

```
<!DOCTYPE HTML><html><head><style type="text/css">
    div{
        width: 230px;
        height: 230px;
        margin: 20px;
        background-color: lightgreen;
    }
</style>
<script>
    function divdobleclic(){
        alert("dio doble clic");
    }
</script></head><body>
    <div id="caja1" ondblclick="divdobleclic()" ></div>
</body></html>
```

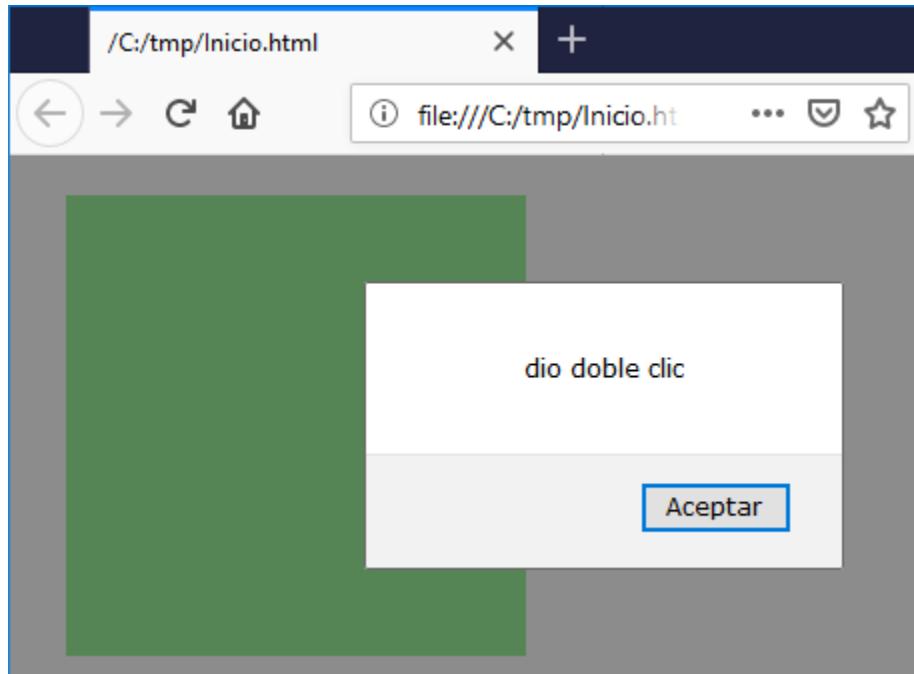


Ilustración 187: Ventana emergente al dar doble clic en un <div>

## Captura el evento de pasar el cursor del ratón por encima de un <div>

Directorio 047

```
<!DOCTYPE HTML><html><head><style type="text/css">
    div{
        width: 230px;
        height: 230px;
        margin: 20px;
        background-color: lightgreen;
    }
</style>
<script>
    function divmouseencima (){
        alert("El ratón pasó por encima");
    }
</script></head><body>
    <div id="caja1" onmouseover="divmouseencima ()" ></div>
</body></html>
```

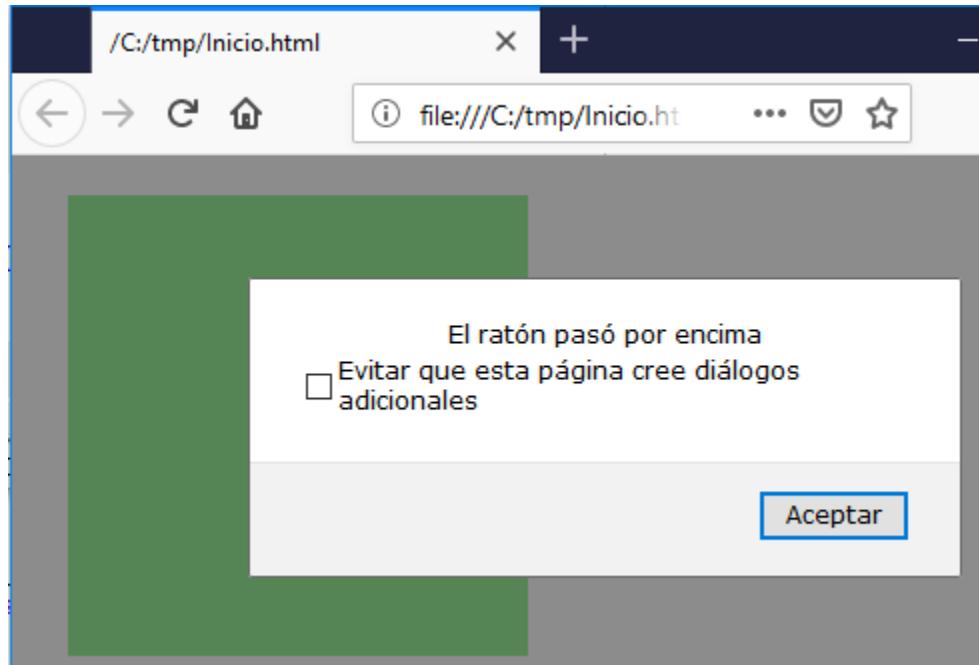


Ilustración 188: Ventana emergente al mover el cursor por encima de un <div>

## Captura el evento de mover el cursor del ratón fuera del <div>

Directorio 047

```
<!DOCTYPE HTML><html><head><style type="text/css">
    div{
        width: 230px;
        height: 230px;
        margin: 20px;
        background-color: lightgreen;
    }
</style>
<script>
    function divmousesale(){
        alert("El ratón ha salido del <div>");
    }
</script></head>
<body>
    <div id="caja" onmouseout="divmousesale()"></div>
</body></html>
```

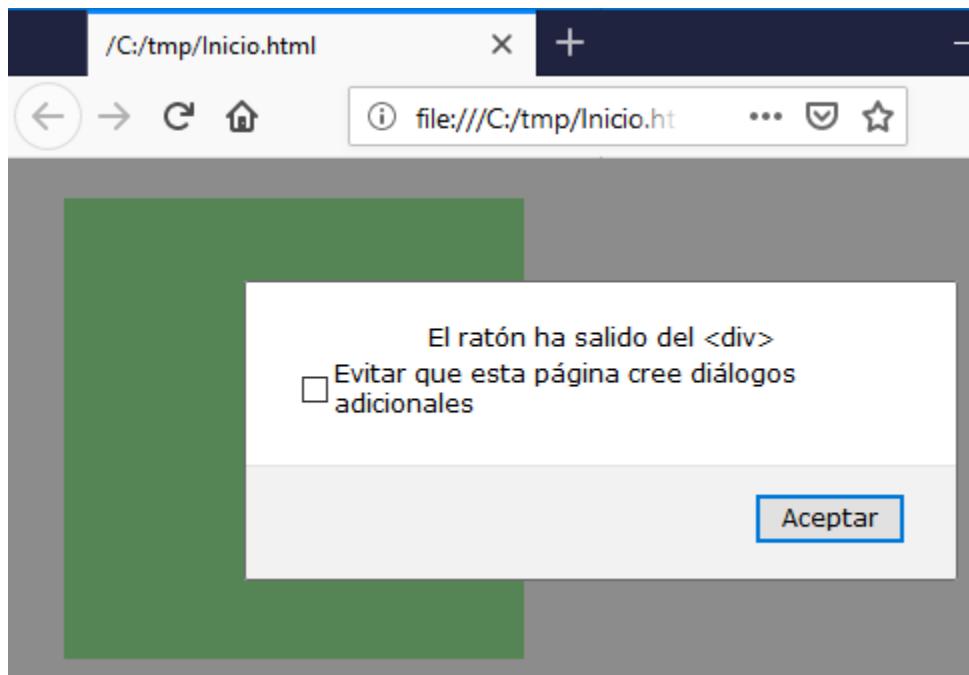


Ilustración 189: Ventana emergente al mover el cursor fuera de un <div>

## Captura el evento apenas cargue la página

Directorio 047

```
<!DOCTYPE HTML><html><head><script>
function cargandopagina() {
    alert("Página se carga");
}
</script></head>
<body onLoad="cargandopagina()">
    <p>Esto es una prueba</p>
</body></html>
```

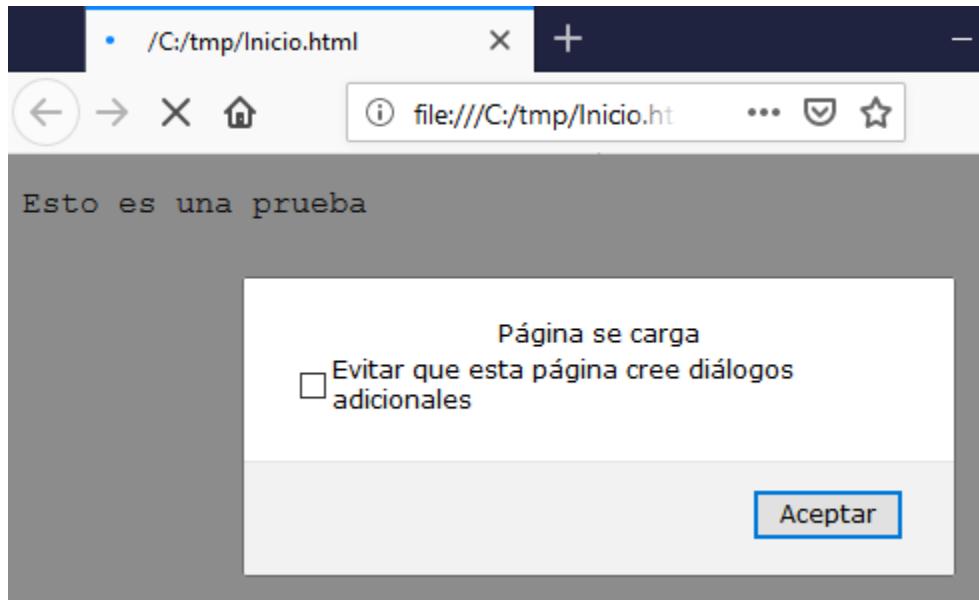


Ilustración 190: Ventana emergente al entrar a una página

## Captura el evento cuando de clic al interior del <div>

Directorio 047

```
<!DOCTYPE HTML><html><head><style type="text/css">
    div{
        width: 230px;
        height: 230px;
        margin: 20px;
        background-color: lightgreen;
    }
</style>
<script>
    function divmousepresiona(){
        alert("clic de ratón");
    }
</script></head><body>
    <div id="caja1" onmousedown="divmousepresiona () " ></div>
</body></html>
```

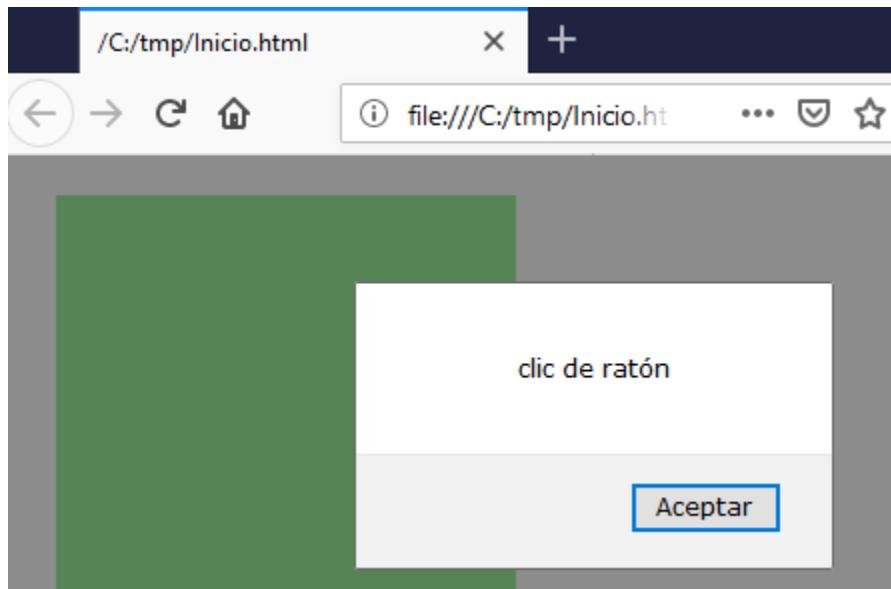


Ilustración 191: Dar clic al interior de un <div>

## Captura el evento al soltar el botón del ratón dentro del <div>

Directorio 047

```
<!DOCTYPE HTML><html><head><style type="text/css">
    div{
        width: 230px;
        height: 230px;
        margin: 20px;
        background-color: lightgreen;
    }
</style>
<script>
    function divmousesuelta(){
        alert("Soltó tecla del ratón");
    }
</script></head><body>
    <div id="caja1" onmouseup="divmousesuelta()"></div>
</body></html>
```

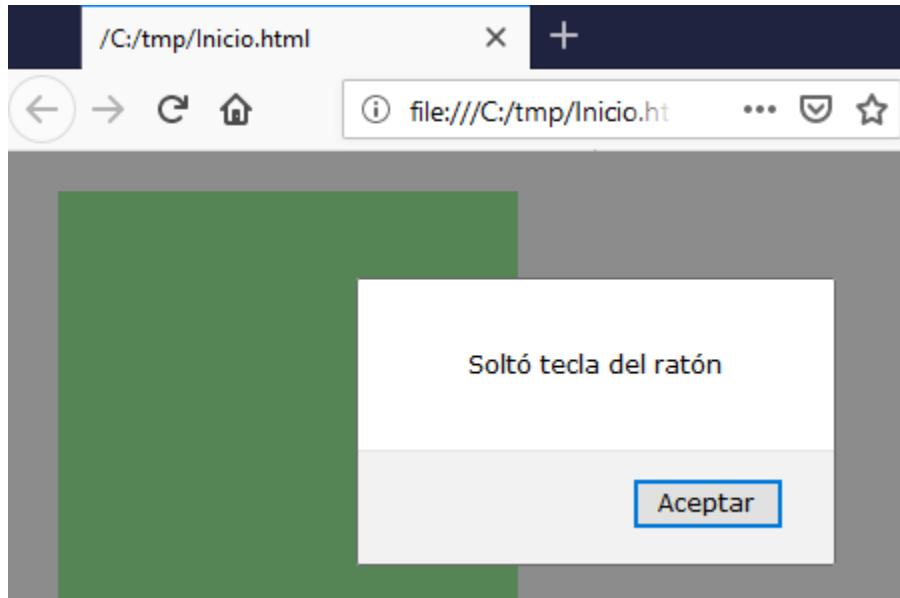


Ilustración 192: Soltar la tecla izquierda del ratón en el interior de un <div>

## Captura el evento de mover el cursor del ratón

Cambiar el color de fondo de la página mientras movemos el cursor del ratón por el <div>

Directorio 047

```
<!DOCTYPE HTML><html><head><style type="text/css">
    div{
        width: 130px;
        height: 130px;
        margin: 20px;
        background-color: lightgreen;
    }
</style>
<script>
function divmousemove() {
    document.bgColor =
' #' +Math.floor(Math.random()*16777215).toString(16);
}
</script></head><body>
    <div id="caja1" onmousemove="divmousemove()"></div>
</body></html>
```



Ilustración 193: Cambiar aleatoriamente el color de fondo al mover el cursor del ratón al interior de un <div>

## Captura el evento cuando se cambia el tamaño de la ventana

Directorio 047

```
<!DOCTYPE HTML><html><head><meta charset="UTF-8"><script>
    function cambialongitud(){
        alert("¡Crecer o decrecer!");
    }
</script></head>
<body onresize="cambialongitud()">
    <p>Cambio el tamaño de la ventana del navegador</p>
</body></html>
```

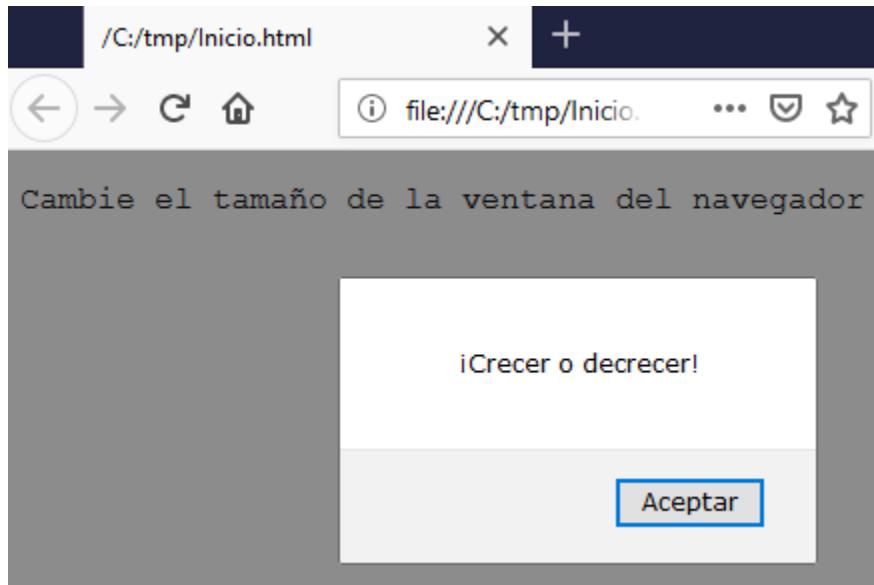


Ilustración 194: Mostrar un aviso cuando se cambia el tamaño de la ventana del navegador

## Captura el evento de dar “submit” en un formulario

Directorio 047

```
<!DOCTYPE HTML><html><head><script>
function formularioenvia() {
    alert("activo envío");
}
</script></head>

<body>
    <form id="formulario" method="post" onsubmit="formularioenvia()">
        <input type="submit" id="boton" value="envía" />
    </form>
</body>
</html>
```

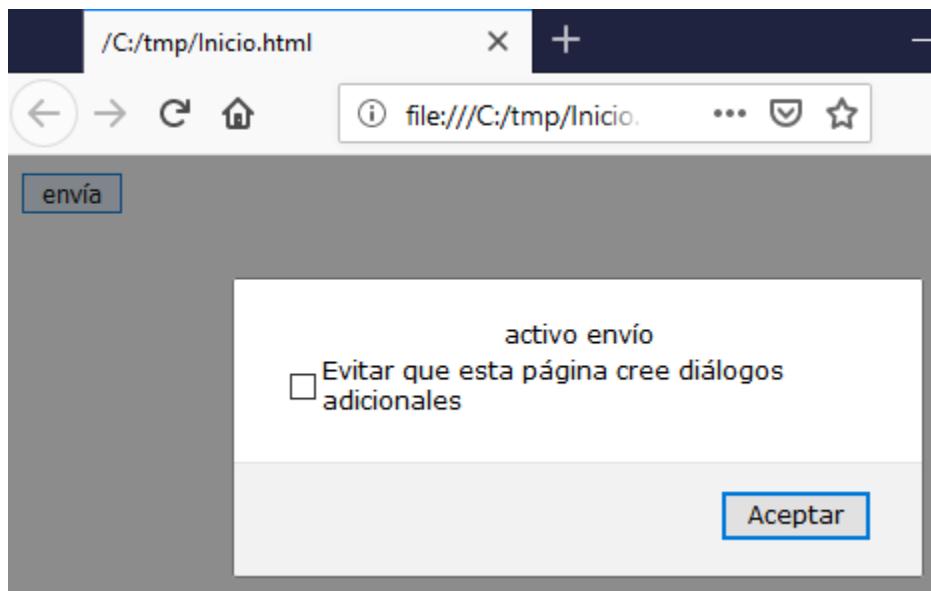


Ilustración 195: Mensaje cuando se presiona botón de envío de formulario

## Captura el evento de entrar a la caja de texto

Directorio 047

```
<!DOCTYPE HTML><html><head>
<script>
function formularioenvia(){ alert("activo envío"); }
function entrafocotexto() { alert("está en la caja de texto"); }
</script></head>

<body>
    <form id="formulario" method="post" onsubmit="formularioenvia()">
        <input type="text" id="texto" onfocus="entrafocotexto()" />
        <input type="submit" id="boton" value="envía" />
    </form>
</body>
</html>
```

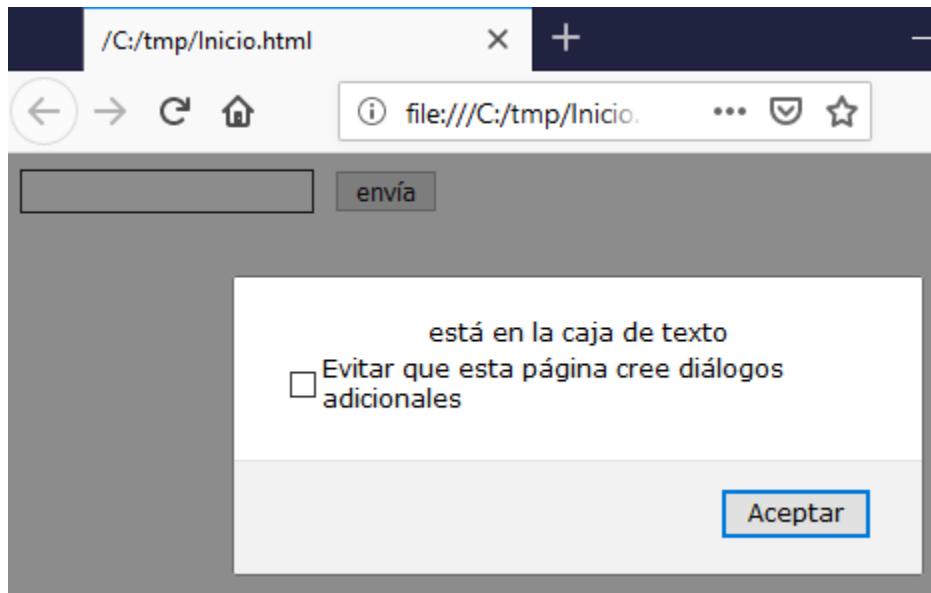


Ilustración 196: Mensaje cuando accede a una caja de texto

## Captura el evento al salir de una caja de texto

Directorio 047

```
<!DOCTYPE HTML><html><head>
<script>
function formularioenvia(){ alert("activo envío"); }
function salefocotexto() { alert("salió de la caja de texto"); }
</script></head>

<body>
    <form id="formulario" method="post" onsubmit="formularioenvia()">
        <input type="text" id="texto" onblur="salefocotexto()" />
        <input type="text" id="segundotexto" />
        <input type="submit" id="boton" value="envía" />
    </form>
</body>
</html>
```

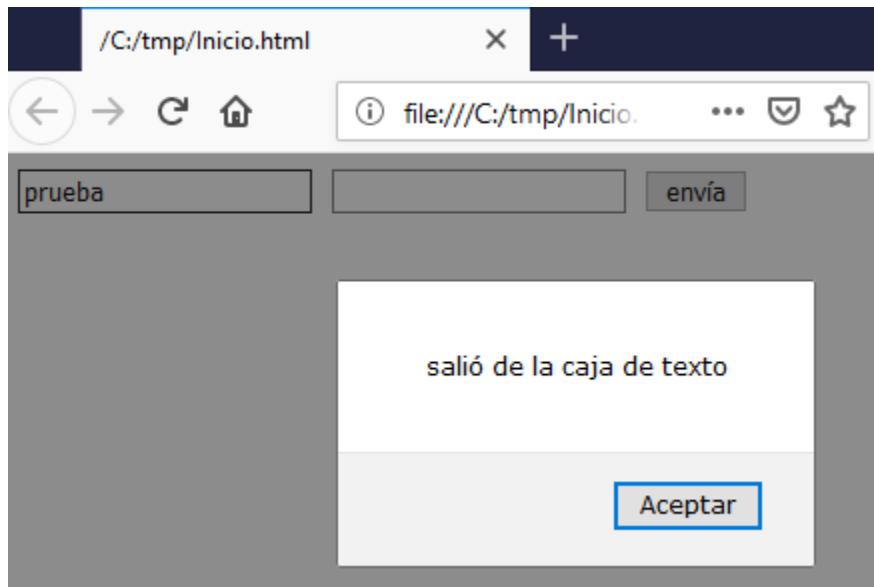


Ilustración 197: Mensaje cuando se sale de una caja de texto

## Captura el evento al presionar una tecla dentro de una caja de texto

Directorio 047

```
<!DOCTYPE HTML><html><head>
<script>
function formularioenvia(){ alert("activo envío"); }
function presionatecla() { alert("Ha presionado una tecla"); }
</script></head>

<body>
    <form id="formulario" method="post" onsubmit="formularioenvia()">
        <input type="text" id="texto" onkeydown="presionatecla()" />
        <input type="submit" id="boton" value="envía" />
    </form>
</body>
</html>
```

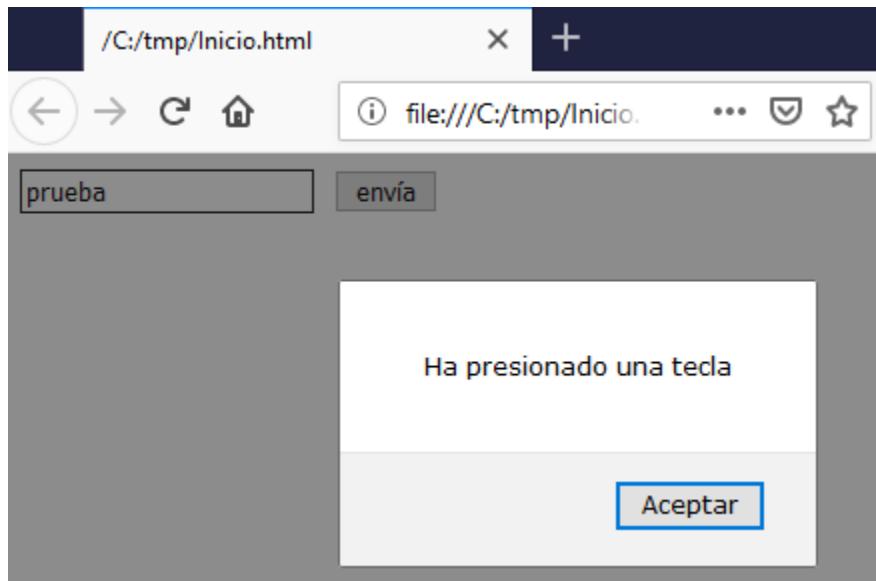


Ilustración 198: Mensaje cuando se presiona cualquier tecla dentro de la caja de texto

Un efecto similar sucede con "onkeypress"

Directorio 047

```
<!DOCTYPE HTML><html><head>
<script>
function formularioenvia(){ alert("activo envío"); }
function presionatecla() { alert("Ha presionado una tecla"); }
</script></head>

<body>
    <form id="formulario" method="post" onsubmit="formularioenvia()">
        <input type="text" id="texto" onkeypress="presionatecla()" />
        <input type="submit" id="boton" value="envía" />
    </form>
</body>
</html>
```

```
</form>
</body>
</html>
```

El evento onkeydown sucede primero, le sigue el evento onkeypress (y este sucede cuando el carácter ha entrado). Podemos probarlo así:

Directorio 047

```
<!DOCTYPE HTML><html><head>
<script>
function formularioenvia(){ alert("activo envío"); }
function abajo() { alert("abajo"); }
function presiona() { alert("presiona"); }
</script></head>

<body>
    <form id="formulario" method="post" onsubmit="formularioenvia()">
        <input type="text" id="texto" onkeypress="presiona()"
onkeydown="abajo()" />
        <input type="submit" id="boton" value="envía" />
    </form>
</body>
</html>
```

## Captura el evento cuando suelta la tecla

Directorio 047

```
<!DOCTYPE HTML><html><head>
<script>
function formularioenvia(){ alert("activo envío"); }
function sueltatecla() { alert("Ha soltado una tecla"); }
</script></head>

<body>
    <form id="formulario" method="post" onsubmit="formularioenvia()">
        <input type="text" id="texto" onkeyup="sueltecla()" />
        <input type="submit" id="boton" value="envía" />
    </form>
</body>
</html>
```

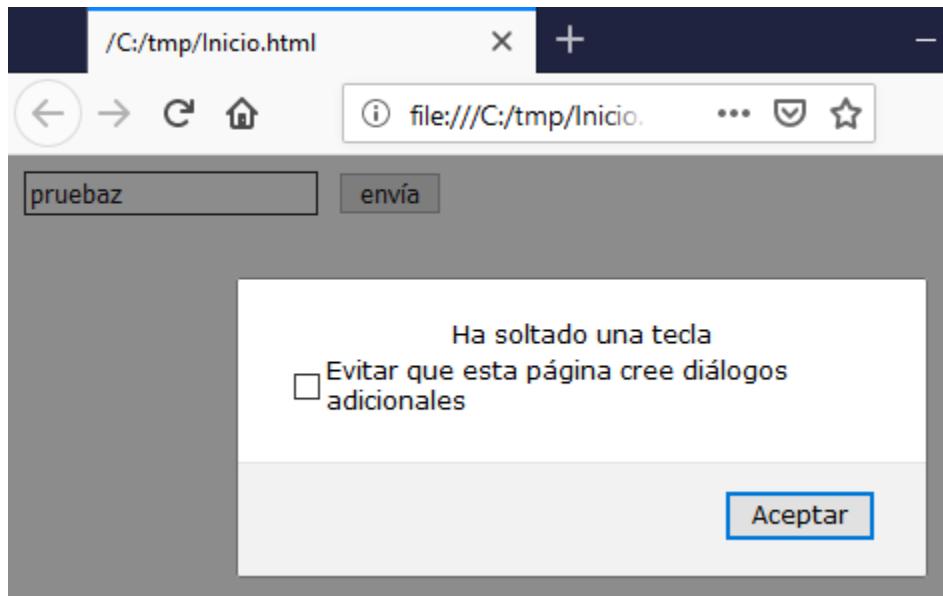


Ilustración 199: Mensaje cuando deja de presionar una tecla en una caja de texto

## Captura el evento de cambio del texto en una caja de texto

Si al saltar a otra caja de texto, la anterior ha cambiado su contenido, se dispara la alerta

Directorio 047

```
<!DOCTYPE HTML><html><head>
<script>
function formularioenvia(){ alert("activo envío"); }
function cambiatexto() { alert("Ha cambiado el texto"); }
</script></head>

<body>
    <form id="formulario" method="post" onsubmit="formularioenvia()">
        <p><input type="text" id="texto" onchange="cambiatexto()" /></p>
        <p><input type="text" id="otro" /></p>
        <p><input type="submit" id="boton" value="envía" /></p>
    </form>
</body>
</html>
```

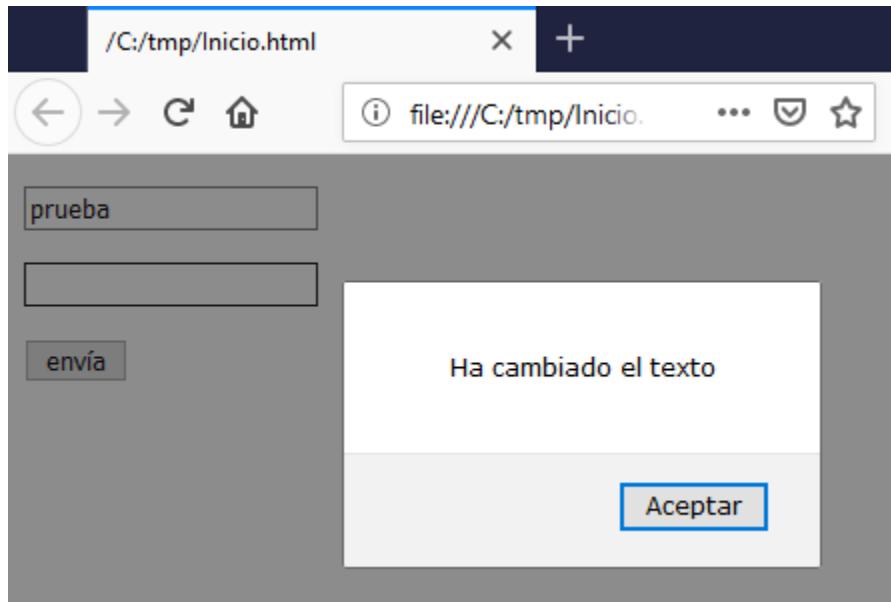


Ilustración 200: Mensaje cuando se ha hecho un cambio dentro de una caja de texto

## Mostrar el código de la tecla presionada

Directorio 047

```
<!DOCTYPE HTML><html><head>
<script>
function formularioenvia(){ alert("envío"); }

//Muestra el código de la tecla presionada
function presionatecla(event) { alert(event.keyCode); }

</script></head>

<body>
    <form id="formulario" method="post" onsubmit="formularioenvia()">
        <input type="text" id="texto" onkeyup="presionatecla(event)" />
        <input type="submit" id="boton" value="envía" />
    </form>
</body>
</html>
```

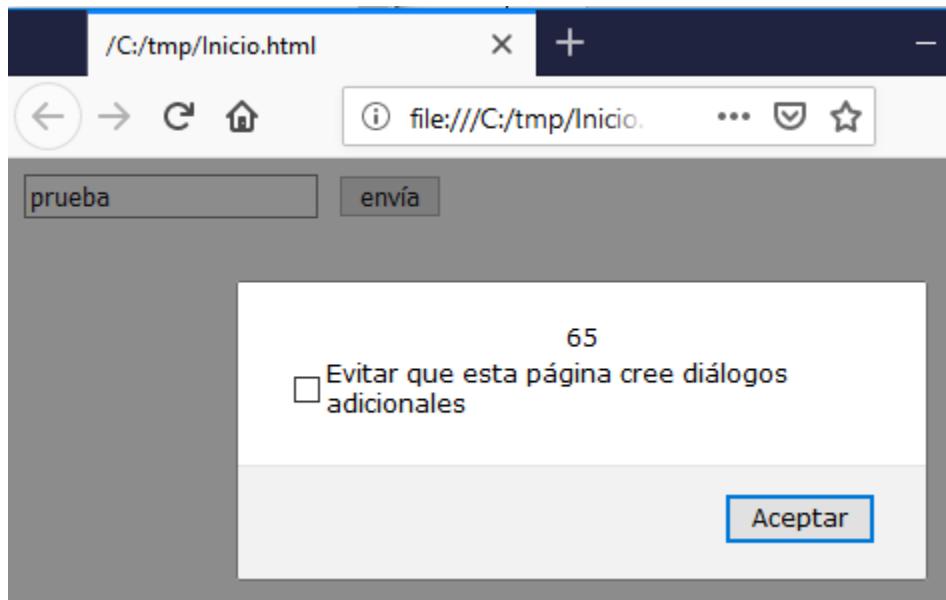


Ilustración 201: Mostrar el código ASCII de la tecla presionada

## Pregunta antes de ir a la página enlazada

Directorio 047

```
<!DOCTYPE HTML><html><head>
<script>
function navegar(){
    return confirm("¿Irá a esa página?");
}
</script></head>

<body>
    <a href="http://darwin.50webs.com" onclick="return
navegar()">Enlace</a>
</body>
</html>
```

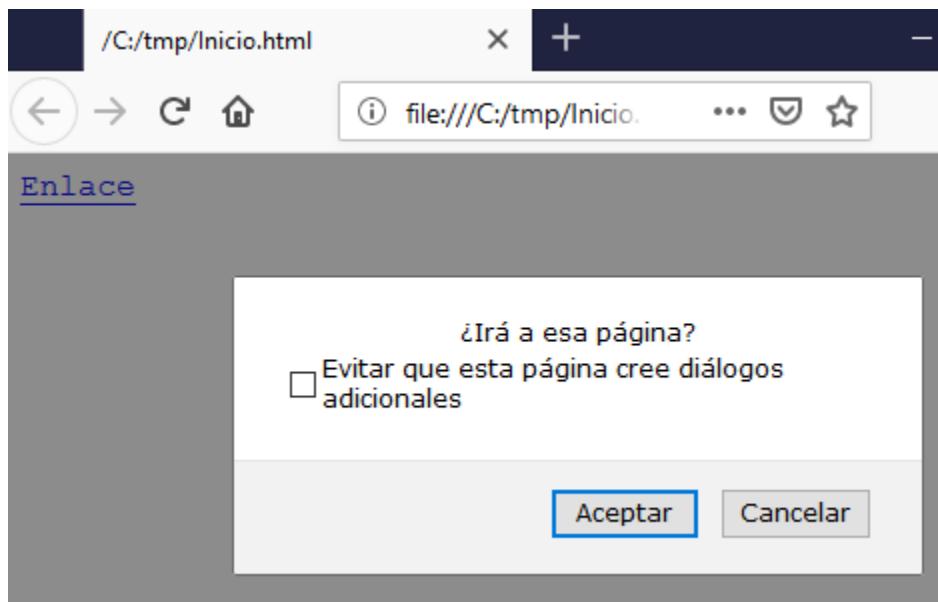


Ilustración 202: Intercepta el poder navegar a otra página

# DOM (Document Object Model)

## Cambiar atributos de un objeto

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Al presionar el botón se llama a esta función:

//Esta función toma el hipervínculo y lo cambia
function enlazarmipagina(){
    //Toma el control del objeto hipervínculo
    var enlace = document.getElementById("ejemploenlace");

    //Cambia hacia donde enlaza
    enlace.href = "http://darwin.50webs.com";
}

</script></head>
<body>
    <a id="ejemploenlace">Hipervínculo</a>
    <input type="button" onclick="enlazarmipagina()" value="Mi página"/>
</body></html>
```

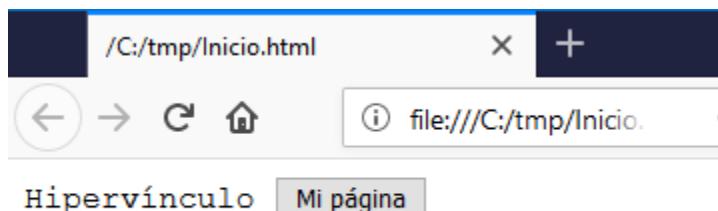


Ilustración 203: Una etiqueta común

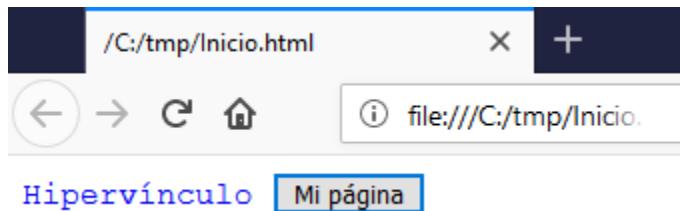


Ilustración 204: La etiqueta común se convierte en hipervínculo

## Cambiar el color de fondo de un <div>

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Cambia el fondo de un <div>
function cambiarfondo() {
    var caja = document.getElementById("cajaejemplo");
    caja.style.backgroundColor = "green";
    caja.style.height = "100px";
    caja.style.width = "50%";
    caja.style.border = "1px solid black";
}
</script></head>
<body>
    <div id = "cajaejemplo" >
        <input type="button" onclick="cambiarfondo()" value="Cambiar
fondo" />
    </div>
</body></html>
```

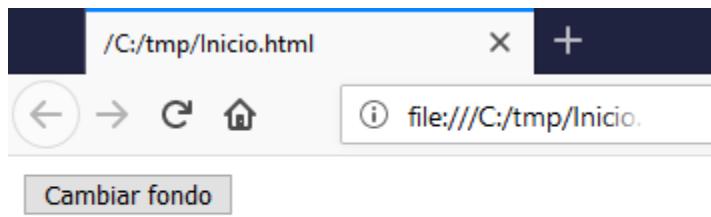


Ilustración 205: Un botón

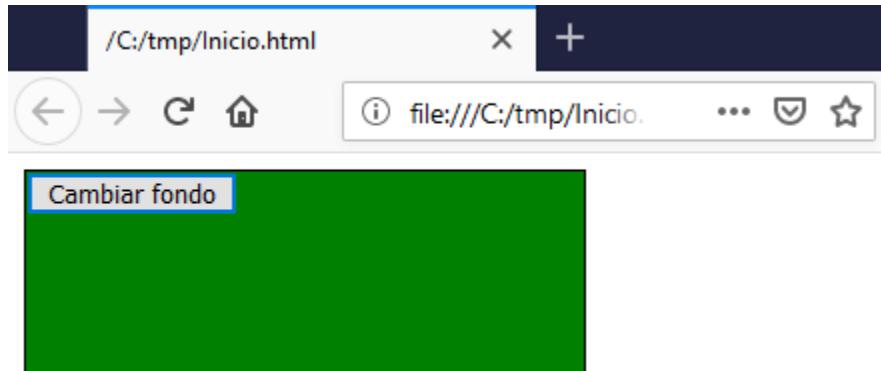


Ilustración 206: Se cambia el color de fondo de un <div> al presionar el botón

## Hacer visible o no un <div>

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Oculta el <div>
function apagar() {
    var caja = document.getElementById("caja");
    caja.style.visibility = "hidden";
}

//Muestra el <div>
function encender() {
    var caja = document.getElementById("caja");
    caja.style.visibility = "visible";
}
</script>
<style type = "text/css">
#caja{
    width: 200px;
    height: 200px;
    background-color: lightgreen;
}
</style></head>
<body>
    <div id="caja"></div>
    <input type="button" value="Apagar" onclick="apagar()" />
    <input type="button" value="Encender" onclick="encender()" />
</body></html>
```



Ilustración 207: Un <div> y dos botones

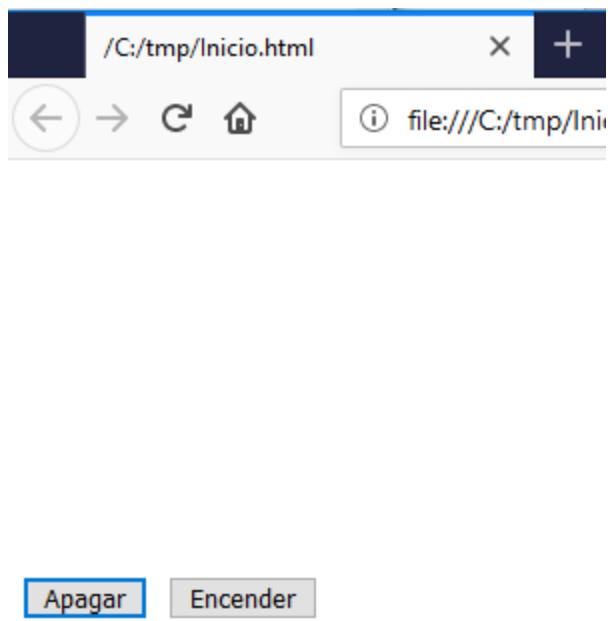


Ilustración 208: Un botón vuelve invisible el <div>

## Poner o quitar texto de un <div>

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Borra contenido del <div>
function borrar(){
    var caja=document.getElementById("caja");
    caja.innerHTML="";
}

//Pone contenido en el <div>
function texto(){
    var caja=document.getElementById("caja");
    caja.innerHTML=<b>dentro del DIV</b>";
}
</script>
<style type = "text/css">
#caja{
    width: 200px;
    height: 200px;
    background-color: lightgreen;
}
</style></head>
<body>
    <div id="caja">Hola Mundo</div>
    <input type="button" value="Borrar" onclick="borrar()" />
    <input type="button" value="Mostrar Texto" onclick="texto()" />
</body></html>
```



Ilustración 209: Un <div> con texto y dos botones

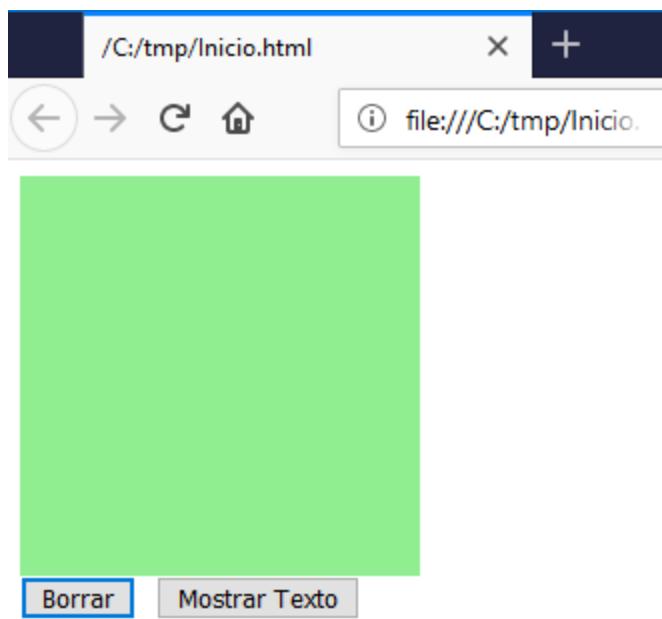


Ilustración 210: Al presionar el botón de "Borrar" se borra el texto del <div>

## Cambiar el aspecto del objeto cuando está seleccionado

Directorio 048

```
<!DOCTYPE HTML><html><head><style type="text/css">
    input{
        background-color: lightgreen;
        border: 1px solid gray;
    }

    input.enfoca {
        background-color: lightblue;
        border: 3px solid green;
    }
</style></head>
<body>
    <input type="text" id="DatoA" onfocus="this.className='enfoca'" 
onblur="this.className=''" />
    <input type="text" id="DatoB" onfocus="this.className='enfoca'" 
onblur="this.className=''" />
</body></html>
```

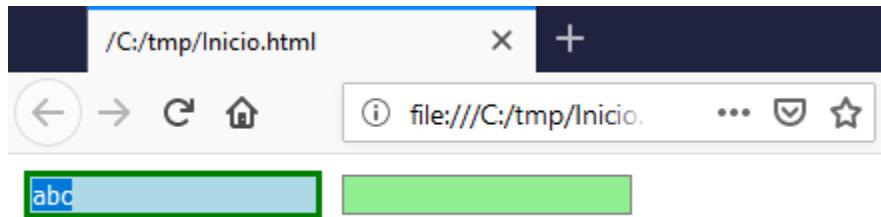


Ilustración 211: Cuando la caja de texto está seleccionada tiene un estilo

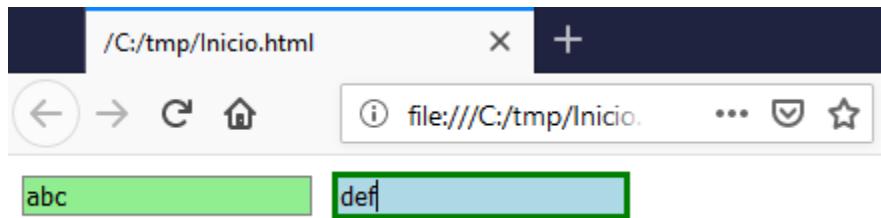


Ilustración 212: Cuando la caja de texto pierde el foco, tendrá otro estilo

Funciona también así:

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
function consiguefoco(entrada){
    entrada.className = 'enfoco';
}

function fueradelfoco(entrada){
```

```
    entrada.className = '';
}
</script><style type = "text/css">
    input{
        background-color: lightgreen;
        border: 1px solid gray;
    }

    input.enfoco {
        background-color: lightblue;
        border: 3px solid green;
    }
</style></head>
<body>
    <input type="text" id="DatoA" onfocus="consiguefoco(this)"
onblur="fueradelfoco(this)" />
    <input type="text" id="DatoB" onfocus="consiguefoco(this)"
onblur="fueradelfoco(this)" />
</body></html>
```

## Validar valores al saltar a otro objeto

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Cuando se salta a otra caja de texto, se valida si
//el dato es un número válido

//Valida si lo que ingresa es un número
function validanumero(entrada){
    if (entrada.value!=""){
        if (isNaN(entrada.value))
            entrada.className='error';
        else
            entrada.className=' ';
    }
}
</script>
<style type="text/css">
input.error{ border: 1px solid red; background-color: yellow; }
</style></head>
<body>
    <label>Valor: <input type="text" id="Numero"
onblur="validanumero(this)" /></label>
    <label> Dato: <input type="text" id="Dato" /></label>
</body></html>
```

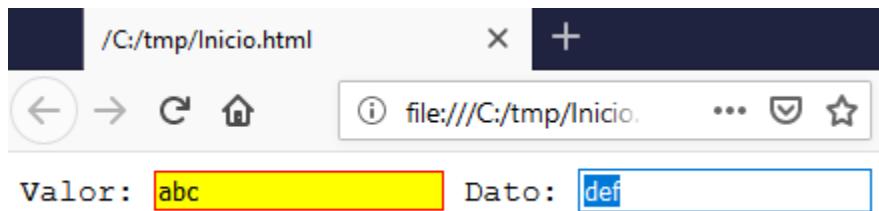


Ilustración 213: Muestra con determinado color si el dato ingresado no es un número válido

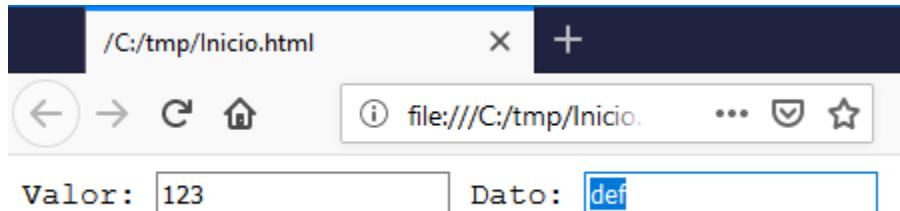


Ilustración 214: Si es un número válido, entonces deja el color por defecto

## Contar objetos

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Hace un conteo de objetos con la etiqueta <p> en la página
function ChequeaObjetos(){
    var obtenerobjetos=document.getElementsByTagName("p");
    alert("Total objetos con <p> : " + obtenerobjetos.length);
}
</script></head>
<body>
<p>Primer párrafo</p>
<p>Segundo párrafo</p>
<p>Tercer párrafo</p>
<input type="button" onclick="ChequeaObjetos()" value="Objetos con <p>">
</body></html>
```

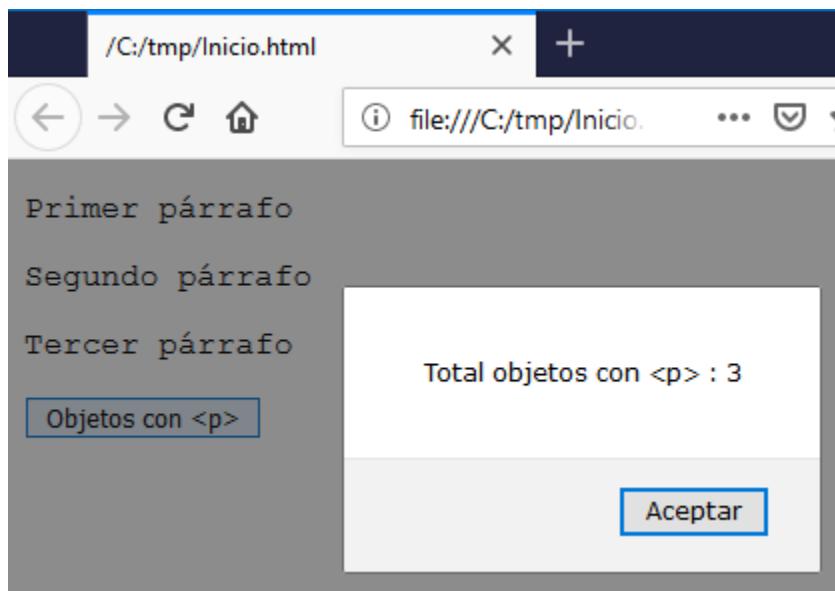


Ilustración 215: Cuenta los objetos con la etiqueta <p>

## Obtener lista de objetos

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Obtiene la lista de objetos con <p>, trae el segundo
//y muestra el código HTML interno
function ChequeaObjetos() {
    var obtenerobjetos=document.getElementsByTagName("p");
    alert("Captura objeto <p> : " + obtenerobjetos[1].innerHTML);
}
</script></head>
<body>
<p>Primer párrafo</p>
<p><b>Segundo párrafo</b></p>
<p><big>Tercer párrafo</big></p>
<input type="button" onclick="ChequeaObjetos()" value="Objetos con <p>">
</body></html>
```

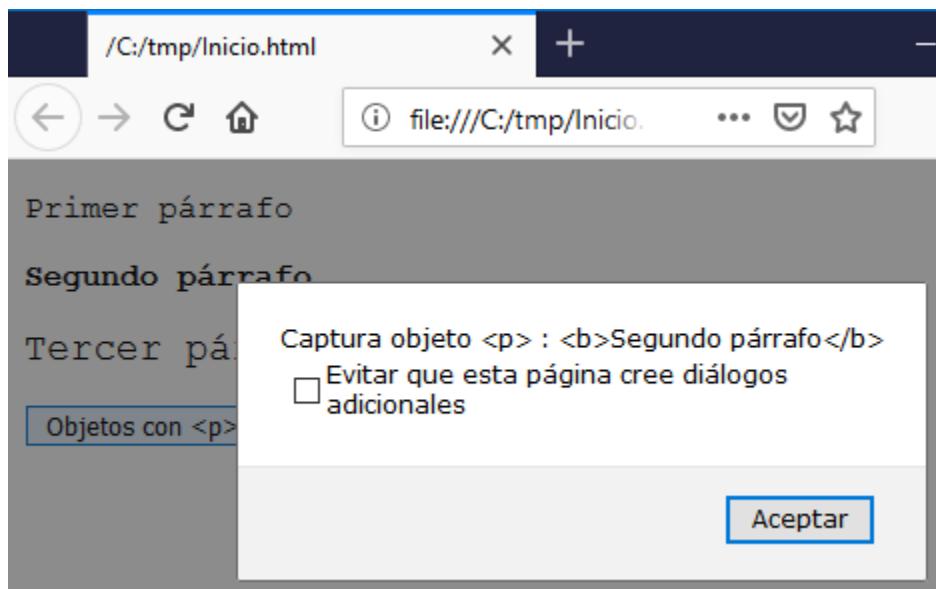


Ilustración 216: Captura un determinado objeto con <p> y lo examina

## Obtener determinado objeto de una lista

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Obtiene la lista de objetos tipo entrada, trae el primero
//y muestra que valor tiene
function ChequeaObjetos() {
    var obtenerobjetos=document.getElementsByTagName("input");
    alert("Valor escrito es : " + obtenerobjetos[0].value);
}
</script></head>
<body>
<input type="text"><br>
<input type="text"><br>
<input type="button" onclick="ChequeaObjetos()" value="Leer valor">
</body></html>
```

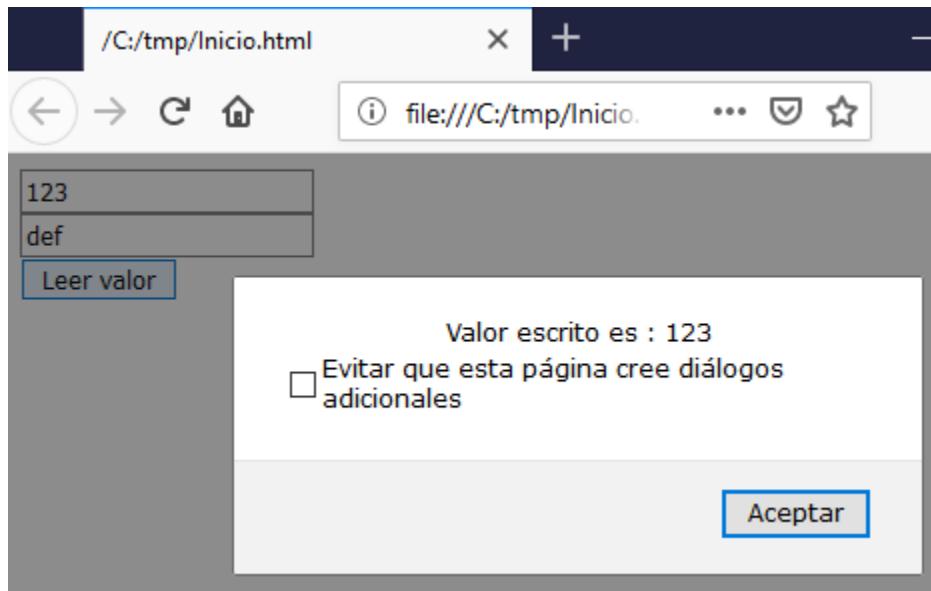


Ilustración 217: Obtiene la lista de objetos tipo entrada, trae el primero y muestra su valor

## Obtener lista de determinado tipo de objetos y ver el valor de sus atributos

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Obtiene la lista de objetos tipo <p>
//y muestra el valor que tiene cada uno
function ChequeaObjetos() {
    var arreglo = new Array();
    var arreglo = document.getElementsByTagName("p");
    for (var cont=0; cont<arreglo.length; cont++)
        alert("Contenido : " + arreglo[cont].innerHTML);
}
</script></head>
<body>
<p>Primer párrafo</p>
<p><b>Segundo párrafo</b></p>
<p>Tercer párrafo</p>
<p>Cuarto párrafo</p>
<input type="button" onclick="ChequeaObjetos()" value="Objetos con <p>">
</body></html>
```

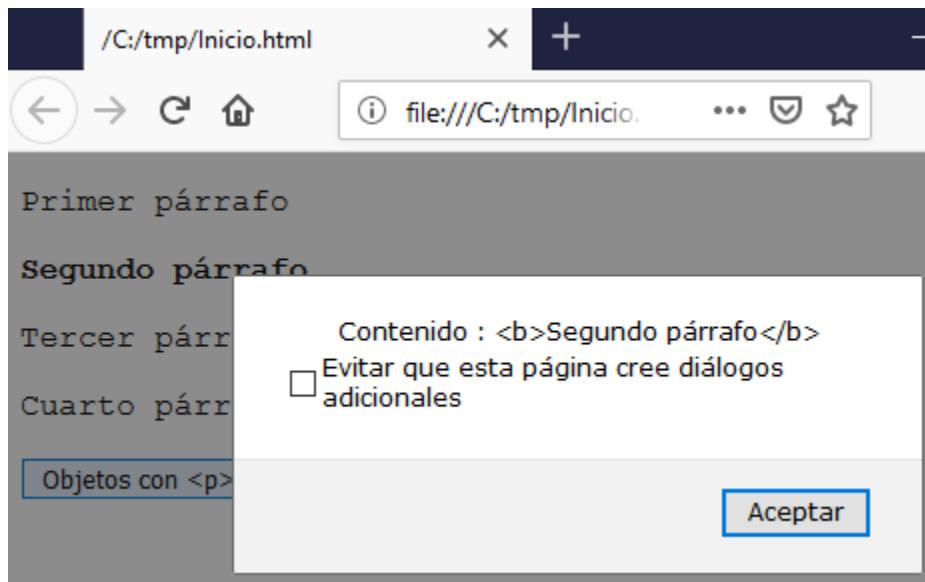
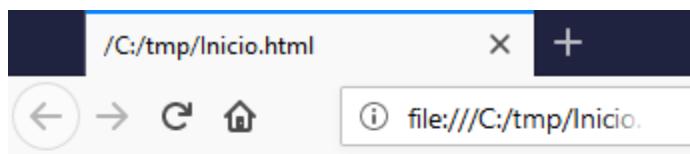


Ilustración 218: Muestra el valor que tiene cada objeto <p>

## Obtener la lista de determinado tipo de objetos y cambiar sus atributos

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Obtiene la lista de objetos tipo <p>
//y cambia la apariencia de cada uno
function CambiaObjetos(){
    var arreglo = new Array();
    var arreglo = document.getElementsByTagName("p");
    for (var cont=0; cont<arreglo.length; cont++)
    { //Cambia la apariencia de cada <p>
        arreglo[cont].style.color = "red";
        arreglo[cont].style["font-family"] = "Impact";
        arreglo[cont].style["font-size"] = "40px";
    }
}
</script></head>
<body>
<p>Primer párrafo</p>
<p>Segundo párrafo</p>
<p>Tercer párrafo</p>
<p>Cuarto párrafo</p>
<input type="button" onclick="CambiaObjetos()" value="Objetos con <p>">
</body></html>
```



Primer párrafo

Segundo párrafo

Tercer párrafo

Cuarto párrafo

Objetos con <p>

Ilustración 219:Párrafos <p> con estilo por defecto

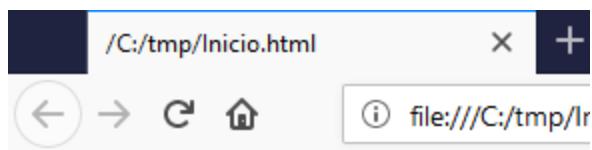


Ilustración 220: Se cambia el estilo de cada objeto <p>

## Mostar la ubicación del documento

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Muestra la ubicación URL de la página
function DarUbicacion(){
    var ubicacion = document.URL;
    alert("URL completa es: " + ubicacion);
}
</script></head>
<body>
<input type="button" onclick="DarUbicacion()" value="¿Donde estoy?">
</body></html>
```

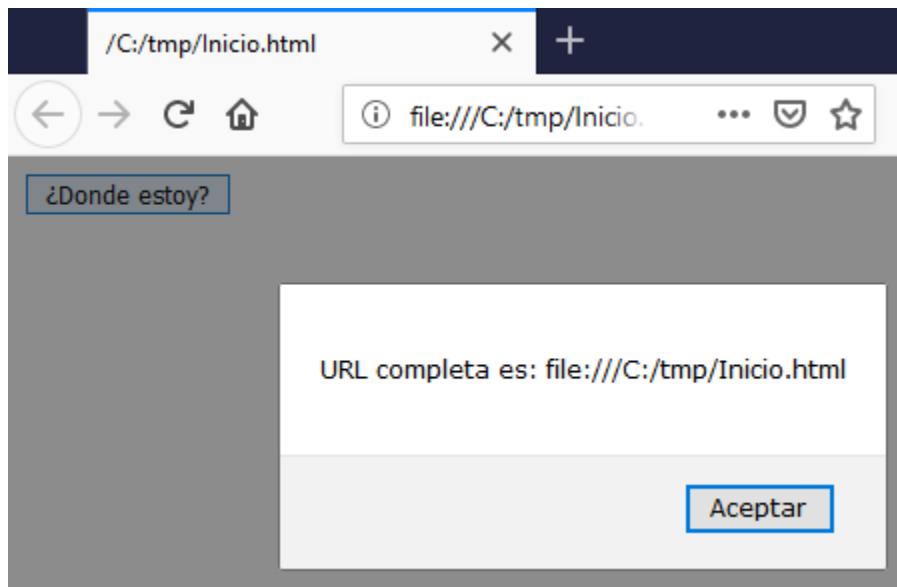
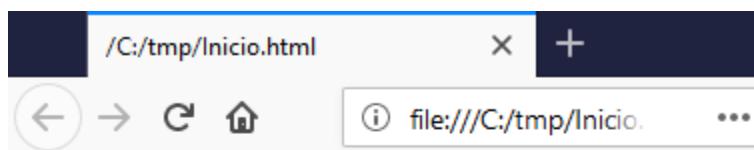


Ilustración 221: Muestra la URL donde se encuentra la página

## Cambiar el estilo CSS de un determinado objeto

Directorio 048

```
<!DOCTYPE HTML><html><head>
<style type="text/css">
.unestilo{ color: blue; font-family: impact; font-size: 50px; }
</style>
<script>
//Aplica un atributo al primer <h1>
function AplicarAtributo(){
    var objetoH1 = document.getElementsByTagName("H1")[0];
    var Atributo = document.createAttribute("class");
    Atributo.value = "unestilo";
    objetoH1.setAttributeNode(Atributo);
}
</script></head>
<body>
<h1>Esta es una prueba</h1>
<h1>Segundo texto</h1>
<button onclick="AplicarAtributo()">Aplicar Atributo</button>
</body></html>
```



**Esta es una prueba**

**Segundo texto**

**Aplicar Atributo**

Ilustración 222: Dos objetos <p>

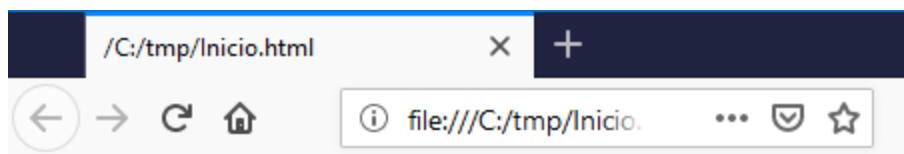


Ilustración 223: Se pueden cambiar los atributos de un objeto <p> en particular

## Mostrar la codificación del documento

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Muestra que codificación tiene la página
function QueCodificacionTiene(){
    var codificacion = document.inputEncoding;
    alert("Codificación es: " + codificacion);
}
</script></head>
<body>
<input type="button" onclick="QueCodificacionTiene()" value="Mi
codificación">
</body></html>
```

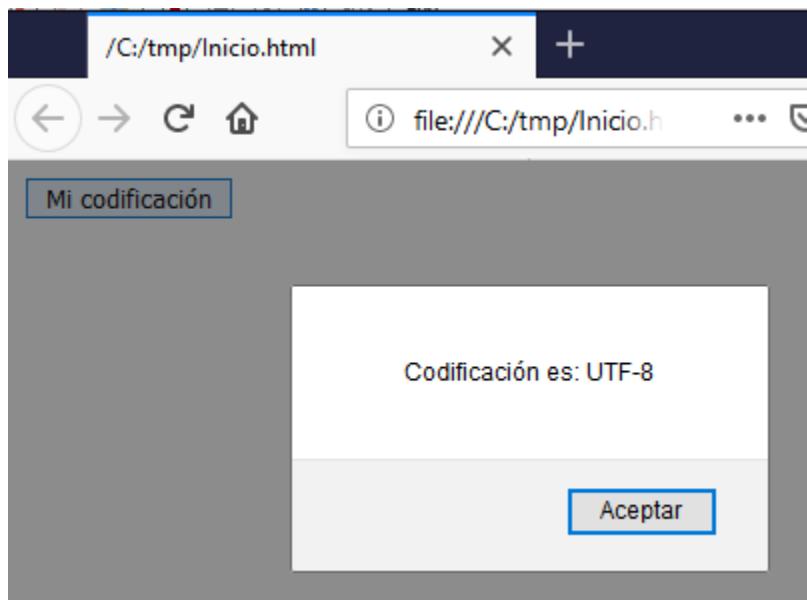


Ilustración 224: Muestra la codificación que tiene la página

## Mostrar el título del documento

Directorio 048

```
<!DOCTYPE HTML><html><head>
<title>Tutorial de JavaScript en Español. Document Object Model
(DOM) .</title>
<script>
//Lee el título de la página
function QueTituloTiene(){
    var titulo = document.title;
    alert("Título es: " + titulo);
}
</script></head>
<body>
<input type="button" onclick="QueTituloTiene()" value="Mi título">
</body></html>
```

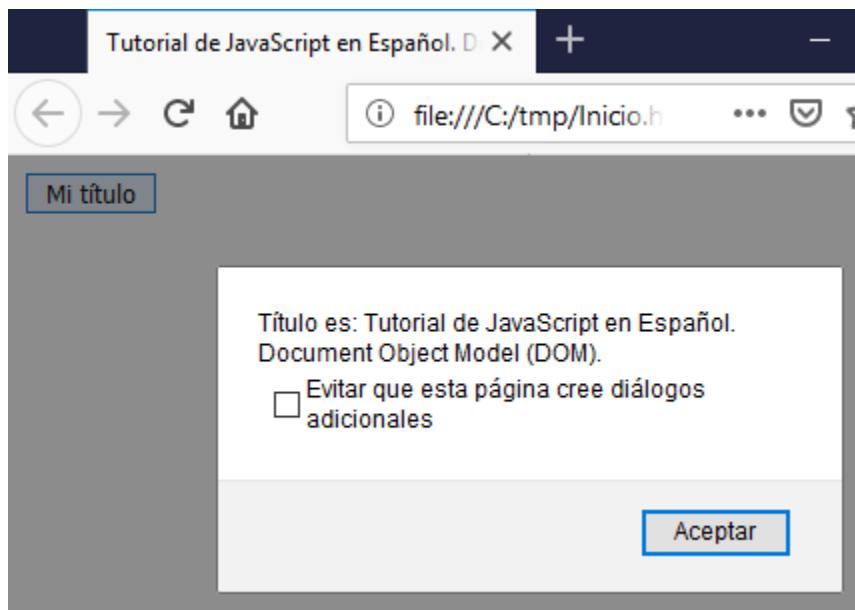
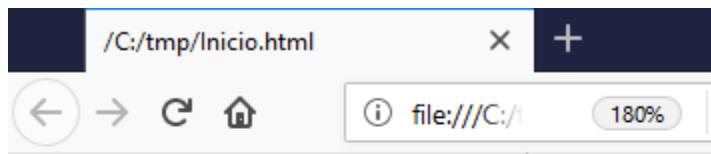


Ilustración 225: Muestra el título que tiene la página

## Saltos de línea sin usar <br>

Directorio 048

```
<!DOCTYPE HTML><html><head></head>
<body>
<pre>
<script>
//Escribe y deja un salto de línea pero requiere usar <pre> </pre>
document.writeln("Esta es una prueba");
document.writeln("de escritura de texto");
document.writeln("en diferentes líneas");
</script>
</pre>
</body></html>
```



Esta es una prueba  
de escritura de texto  
en diferentes líneas

Ilustración 226: Saltos de línea sin usar <br>

## Mostrar la fecha de modificación del documento

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Muestra la fecha de modificación del documento
function Modificacion(){
    var fecha = document.lastModified;
    alert("Este documento fue modificado: " + fecha);
}
</script></head>
<body>
<input type="button" onclick="Modificacion()" value="¿Última
modificación?">
</body></html>
```

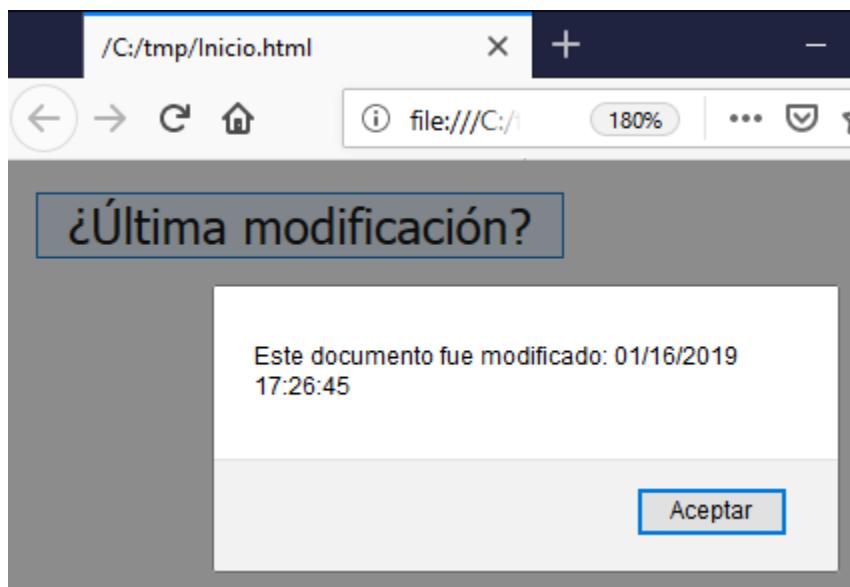


Ilustración 227: Muestra la fechas de modificación de la página

## Mostrar el tipo de documento

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Muestra el tipo de documento
function Tipo(){
    var tipodocumento = document.doctype.name;
    alert("Tipo de documento: " + tipodocumento);
}
</script></head>
<body>
<input type="button" onclick="Tipo()" value="¿Tipo de documento?">
</body></html>
```

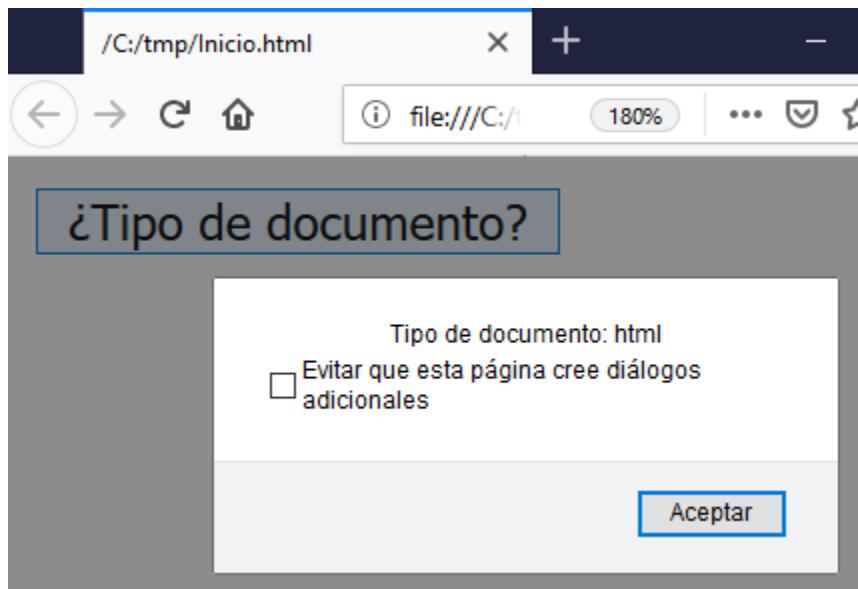


Ilustración 228: Muestra el tipo de documento

## Mostrar el dominio del documento

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Muestra el dominio donde está el documento
function DominioDocumento() {
    var dominio = document.domain;
    alert("Dominio: " + dominio);
}
</script></head>
<body>
<input type="button" onclick="DominioDocumento()" value="¿Dominio?">
</body></html>
```

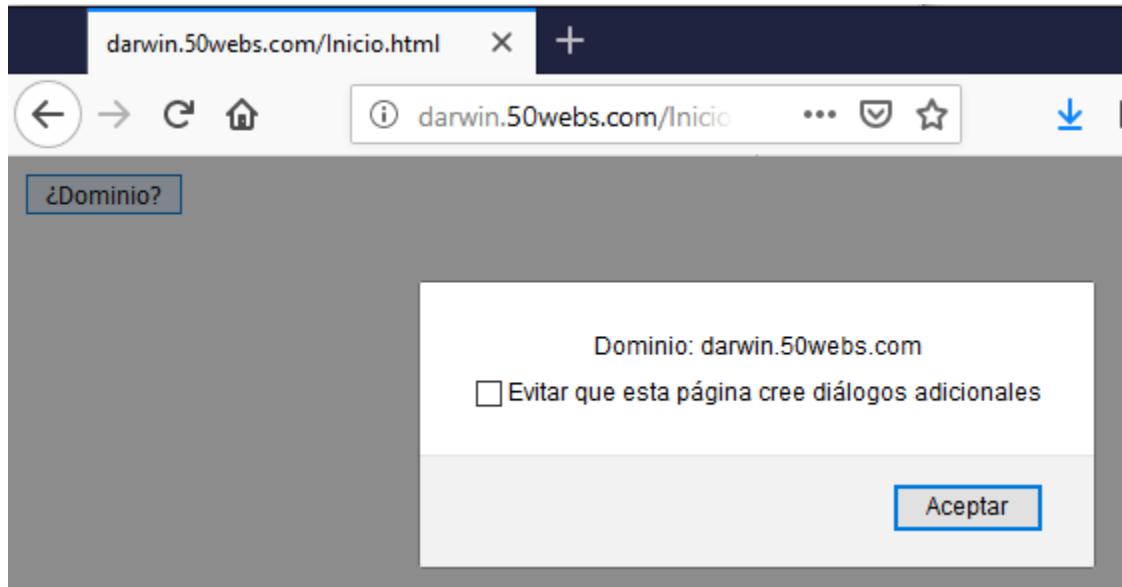


Ilustración 229: Mostrar el dominio donde está alojada la página

## Mostrar el estado del documento

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Muestra el estado del documento
function Estado() {
    var estadodocumento = document.readyState;
    alert("Estado del documento: " + estadodocumento);
}
</script></head>
<body>
<input type="button" onclick="Estado()" value="¿Está listo?">
</body></html>
```

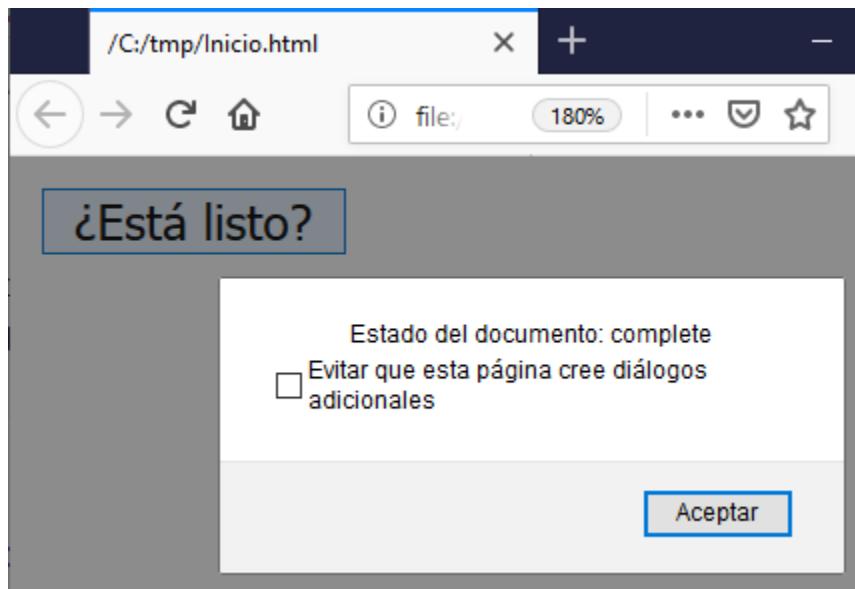


Ilustración 230: Muestra el estado del documento. Si ha cargado completamente es "complete"

## Mostrar las dimensiones del documento

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Muestra las dimensiones del documento
//Nota: Al agregar mas líneas con <p> aumenta la altura
function Dimensiones() {
    var altura = document.body.clientHeight;
    var ancho = document.body.clientWidth;
    alert("Altura: " + altura + " Ancho: " + ancho);
}
</script></head>
<body>
<input type="button" onclick="Dimensiones()" value="Tamaños">
<p>Prueba</p>
</body></html>
```

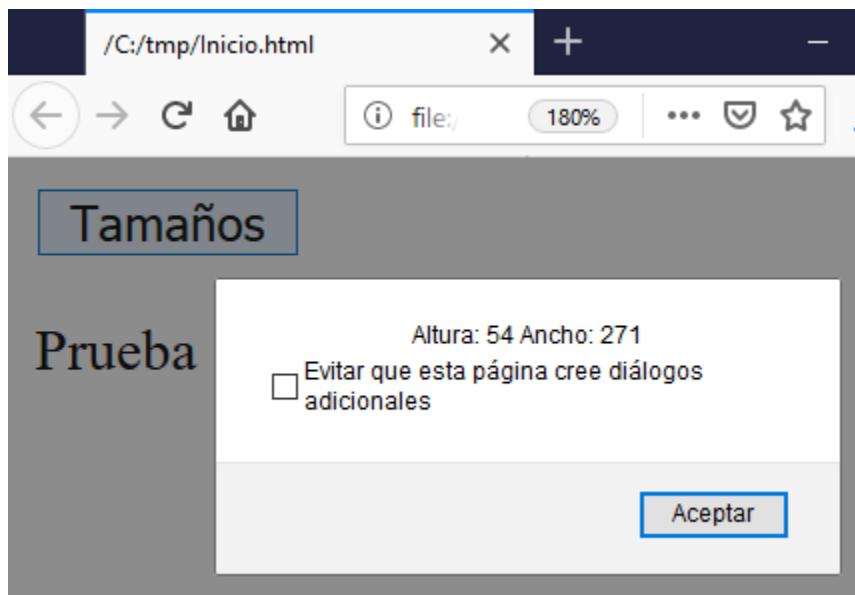


Ilustración 231: Alto y ancho de la página

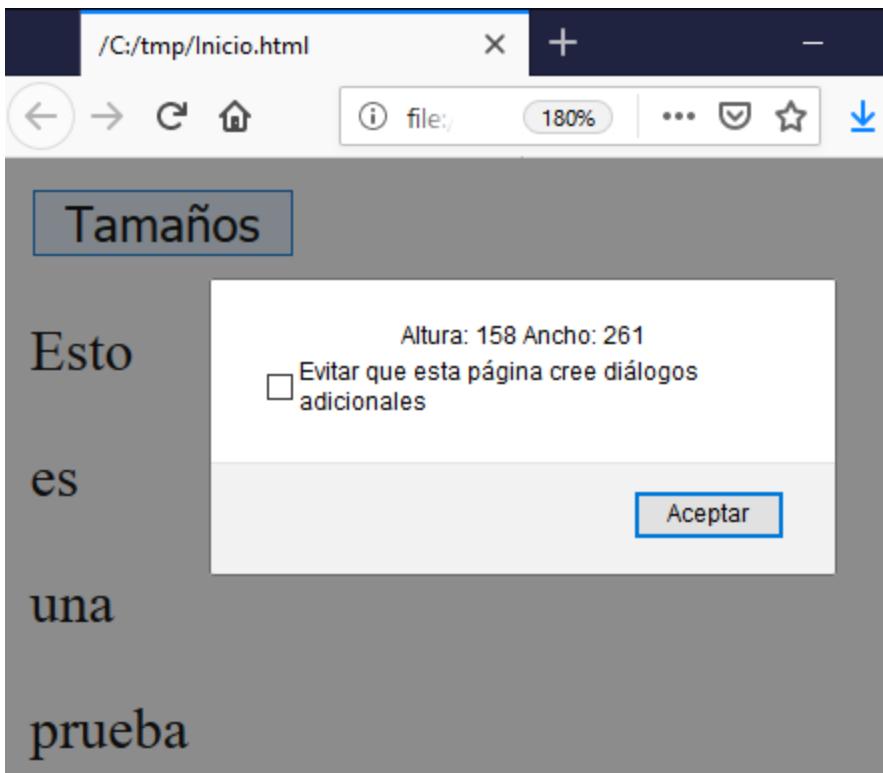
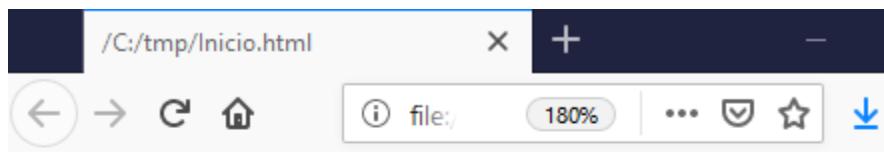


Ilustración 232: Alto y ancho de la página adicionando más <p>

## Mostrar el número de enlaces

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Cuenta el número de hipervínculos que hay en el documento
function TotalEnlaces(){
    alert("Enlaces: " + document.links.length);
}
</script></head>
<body>
<a href="http://darwin.50webs.com">Mi página web</a><br>
<a href="https://openlibra.com/es/book/redes-neuronales-parte-1">Libro de
Redes Neuronales</a><br>
<input type="button" onclick="TotalEnlaces ()" value="¿Enlaces?">
</body></html>
```



Mi página web  
Libro de Redes Neuronales  
¿Enlaces?

Ilustración 233: Una página con dos enlaces

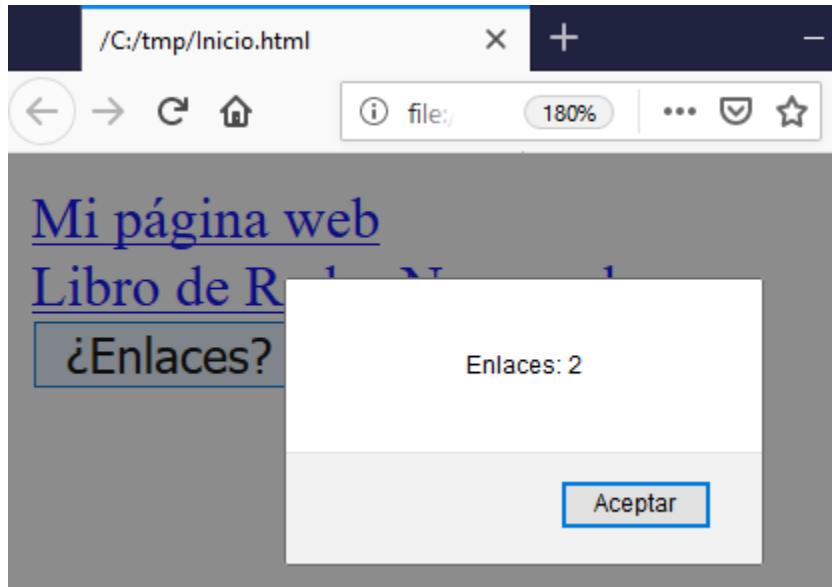
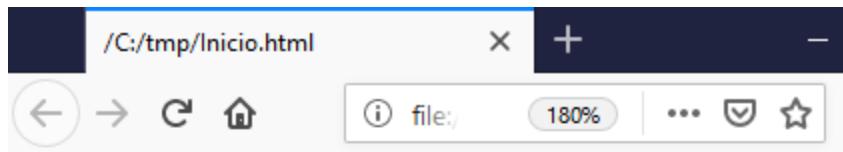


Ilustración 234: Cuenta el número de enlaces de la página

## Mostrar hacia donde apuntan los enlaces

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Muestra cada hipervínculo que hay en el documento
function MuestraEnlaces(){
    for(var cont=0; cont<document.links.length; cont++)
        alert("Enlace: " + document.links[cont].href);
}
</script></head>
<body>
<a href="http://darwin.50webs.com">Mi página web</a><br>
<a href="https://openlibra.com/es/book/redes-neuronales-parte-1">Redes
neuronales</a><br>
<a href="https://openlibra.com/es/book/evaluador-de-expresiones-
algebraicas-ii">Evaluador de expresiones</a><br>
<input type="button" onclick="MuestraEnlaces()" value="¿Enlaces?">
</body></html>
```



Mi página web  
Redes neuronales  
Evaluador de expresiones  
¿Enlaces?

<https://openlibra.com/es/book/evaluador-de-expresiones-algebraicas-ii>

Ilustración 235: Una página con tres enlaces

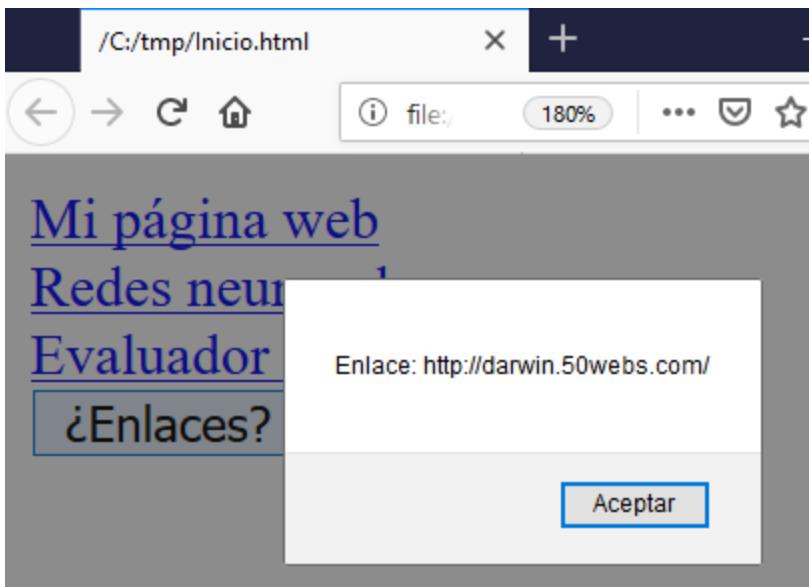


Ilustración 236: Muestra el enlace de cada página

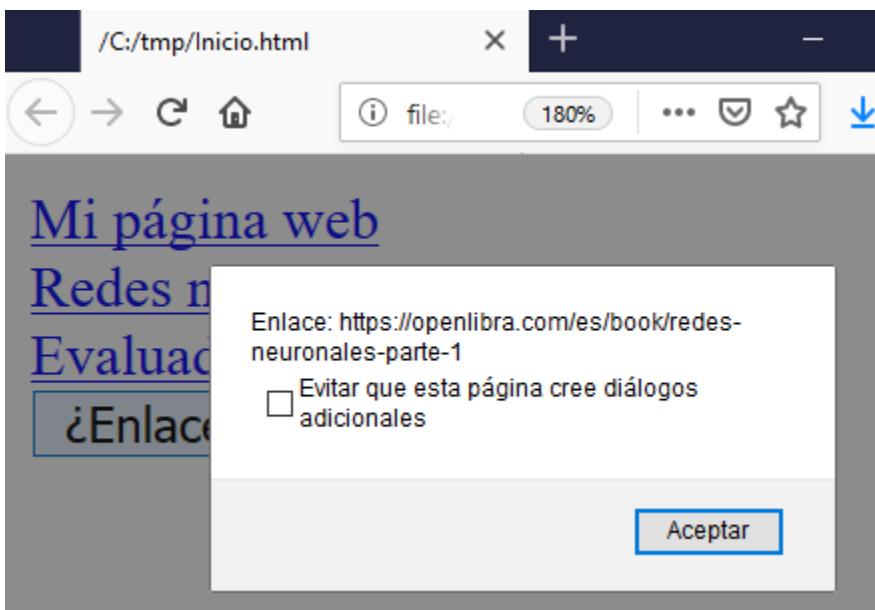


Ilustración 237: Muestra el enlace de cada página

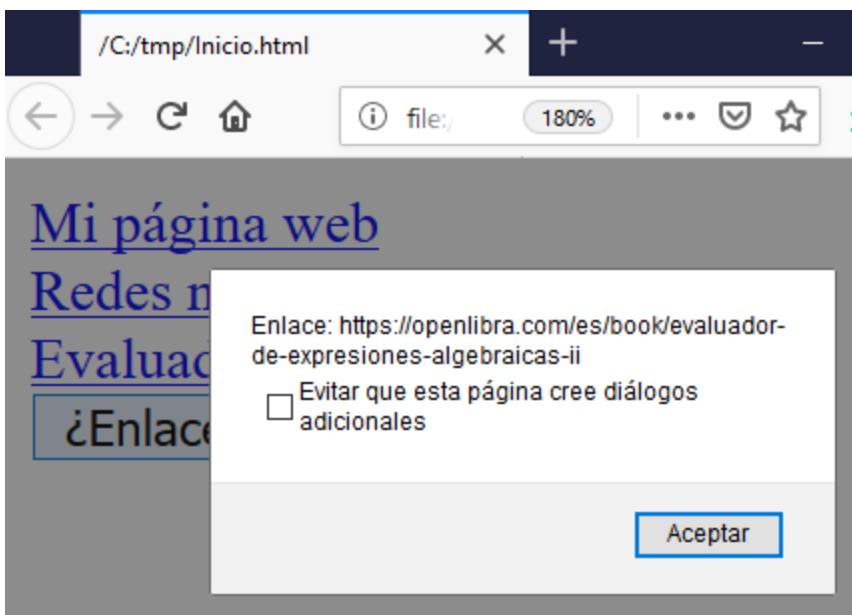


Ilustración 238: Muestra el enlace de cada página

## Crear botones en tiempo de ejecución

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Crea controles (botones) en tiempo de ejecución
function CreaBotones() {
    //Genera un botón
    var boton = document.createElement("button");

    //Genera un texto
    var texto = document.createTextNode("Presionar");

    //Agrega el texto al botón
    boton.appendChild(texto);

    //Agrega el botón al <body>
    document.body.appendChild(boton);
}
</script></head>
<body>
<input type="button" onclick="CreaBotones()" value="Generar Botones">
</body></html>
```

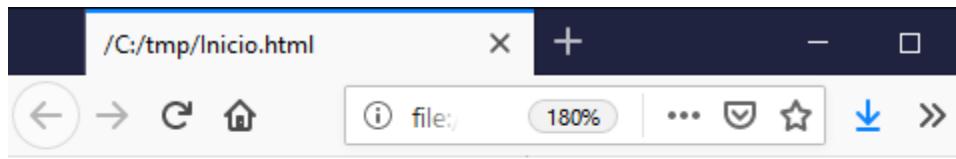


Ilustración 239: Un documento con un botón

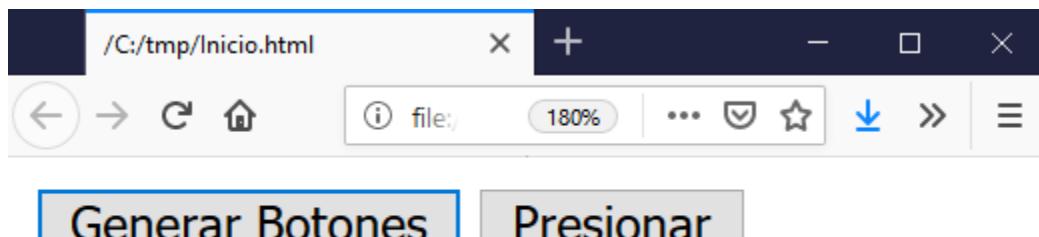


Ilustración 240: Al presionar el botón se genera un nuevo botón

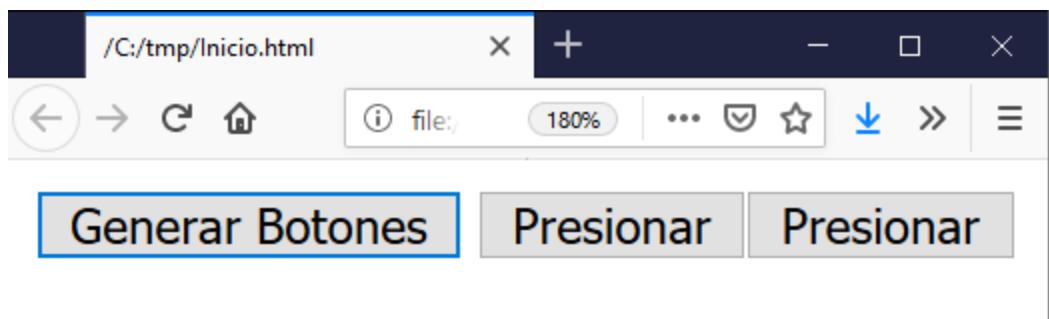


Ilustración 241: Y se pueden generar más botones en tiempo de ejecución

## Crear <p> en tiempo de ejecución

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Crea etiquetas en tiempo de ejecución
function CreaEtiquetas(){
    //Genera una etiqueta
    var etiqueta = document.createElement("H1");

    //Genera un texto
    var texto = document.createTextNode("Este es un texto");

    //Agrega el texto a la etiqueta
    etiqueta.appendChild(texto);

    //Agrega la etiqueta al <body>
    document.body.appendChild(etiqueta);
}
</script></head>
<body>
<input type="button" onclick="CreaEtiquetas()" value="Generar Etiquetas">
</body></html>
```

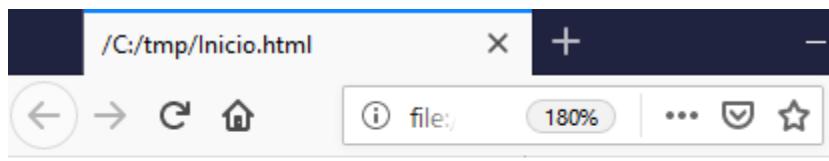
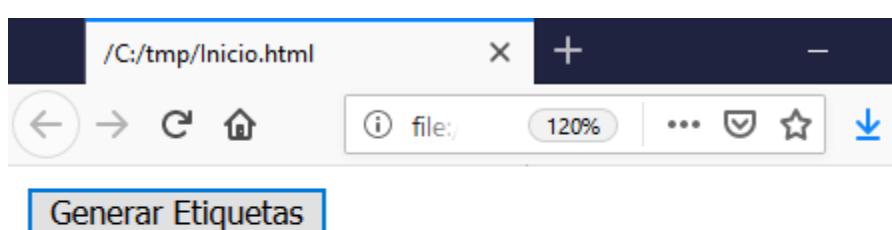
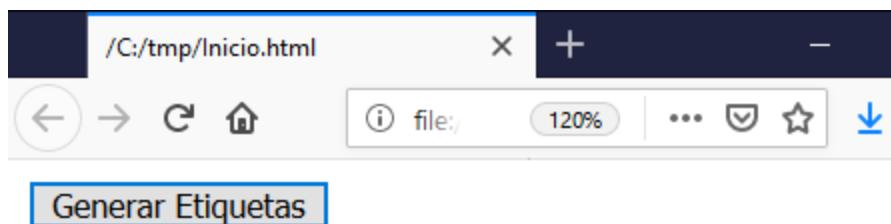


Ilustración 242: Un documento con un botón



# Este es un texto

Ilustración 243: Al presionar el botón se genera un nuevo <p>



**Este es un texto**

**Este es un texto**

*Ilustración 244: Se generan nuevos <p> en tiempo de ejecución*

## Crear nuevos <p> y contarlos en tiempo de ejecución

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Crea etiquetas en tiempo de ejecución
function CreaEtiquetas(){
    var etiqueta = document.createElement("H1");
    var texto = document.createTextNode("Este es un texto");
    etiqueta.appendChild(texto);
    document.body.appendChild(etiqueta);
}

//Cuenta el número de etiquetas que hay en el documento
function TotalEtiquetas(){
    var etiquetas = document.getElementsByTagName("H1");
    alert("Total etiquetas: " + etiquetas.length);
}
</script></head>
<body>
<input type="button" onclick="CreaEtiquetas()" value="Generar Etiquetas">
<input type="button" onclick="TotalEtiquetas()" value="Total Etiquetas">
</body></html>
```

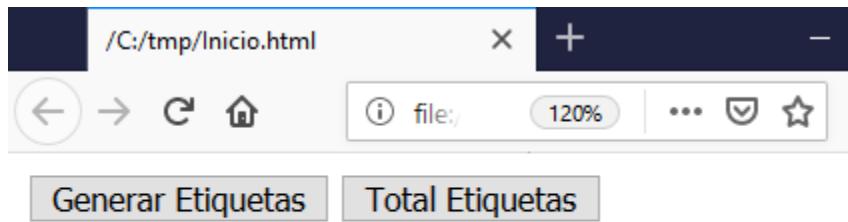
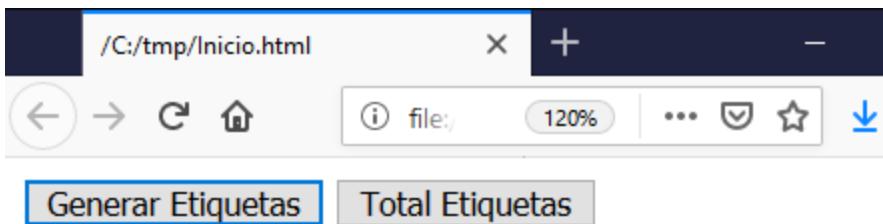


Ilustración 245: Un documento con dos botones



Este es un texto

Este es un texto

Ilustración 246: Se generan dos <p> en tiempo de ejecución

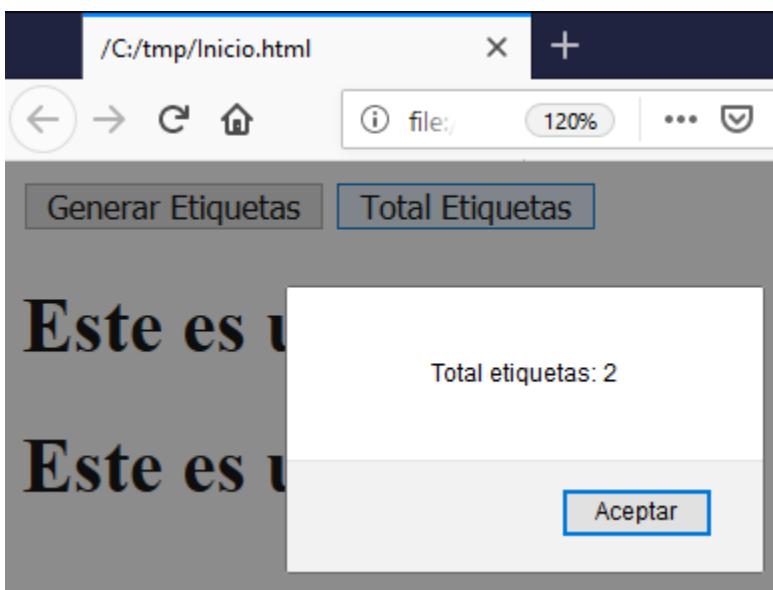


Ilustración 247: Se cuenta el número de <p> en el documento

## Contar el número de objetos en el documento

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Trae el número de elementos con un determinado "name"
function TotalElementos() {
    var elementosNombre = document.getElementsByName("prueba");
    alert("Número de elementos: " + elementosNombre.length);
}
</script></head><body>
<input name="prueba" type="text" value="valor A"><br>
Selección: <input name="prueba" type="radio" value="valor B"><br>
Botón: <input name="prueba" type="button" value="valor C"><br>
<input type="button" onclick="TotalElementos()" value="¿Total Elementos?">
</body></html>
```

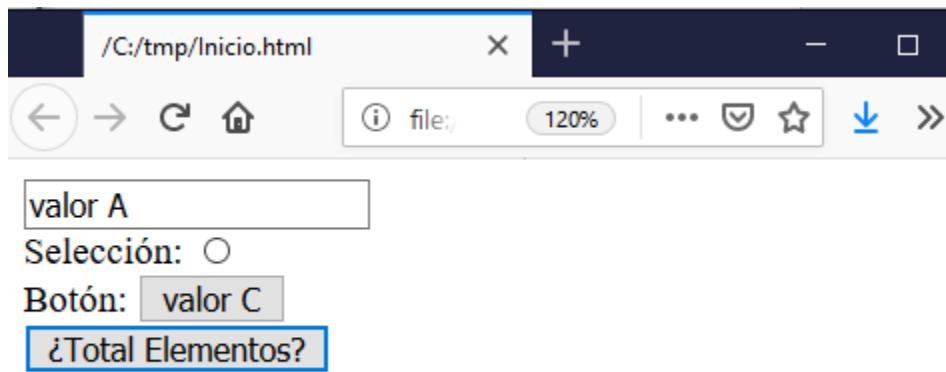


Ilustración 248: Varios tipos de objeto en la página

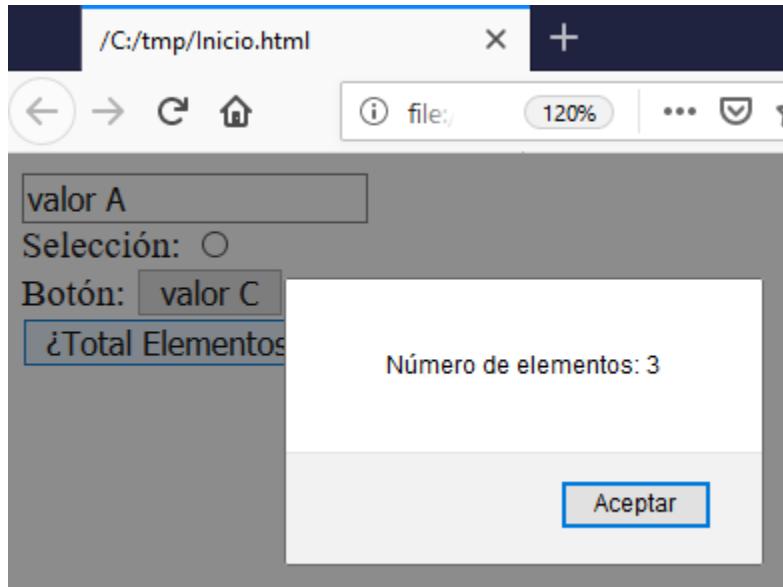


Ilustración 249: Cuenta el número de objetos en el documento

## Contar el número de formularios en el documento

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Trae el número de formularios
function TotalFormularios(){
    alert("Total formularios: " + document.forms.length);
}
</script></head>
<body>
<form name="Form1"></form>
<form name="Form2"></form>
<form></form>
<input type="button" onclick="TotalFormularios()" value="¿Total
Formularios?">
</body></html>
```

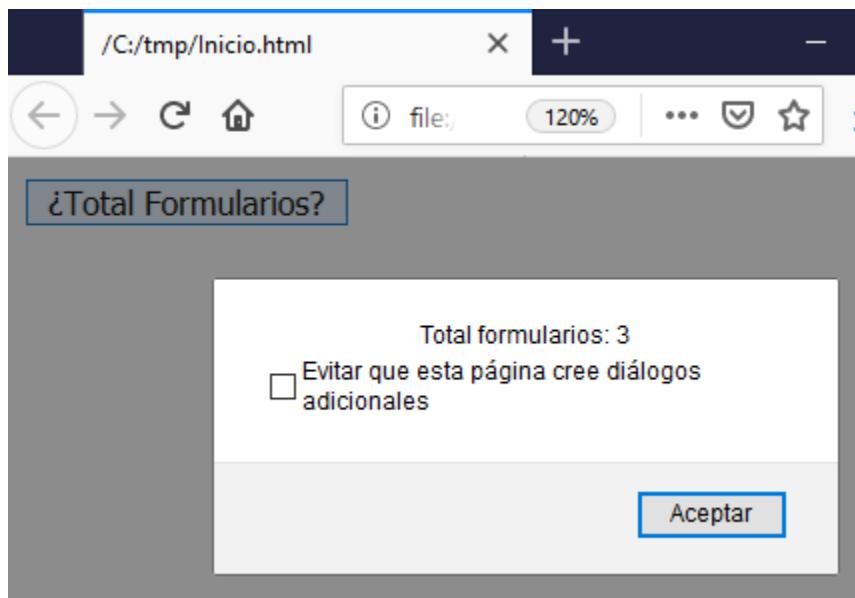


Ilustración 250: Cuenta el número de formularios en el documento

## Contar el número de imágenes del documento

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Trae el total de imágenes en el documento
function TotalImagenes(){
    alert("Total imágenes: " + document.images.length);
}
</script></head>
<body>
</img><br>
<input type="button" onclick="TotalImagenes()" value="¿Total imágenes?">
</body></html>
```

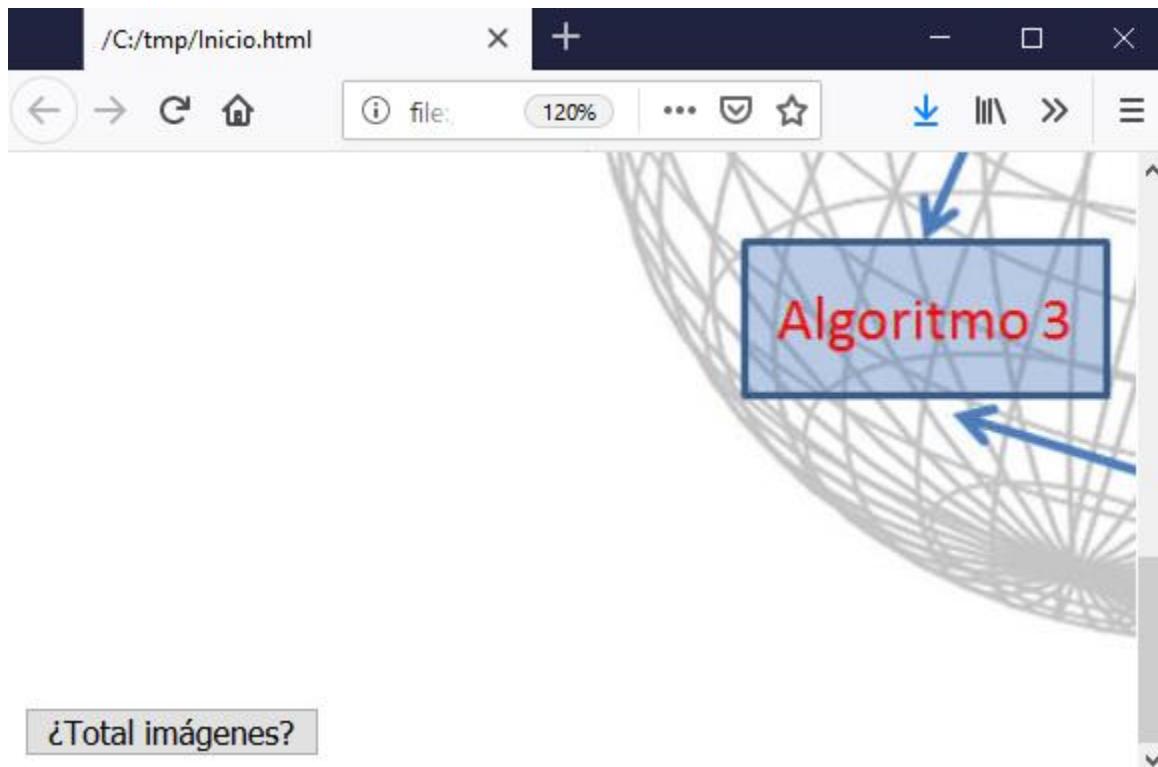


Ilustración 251: Un documento con una imagen

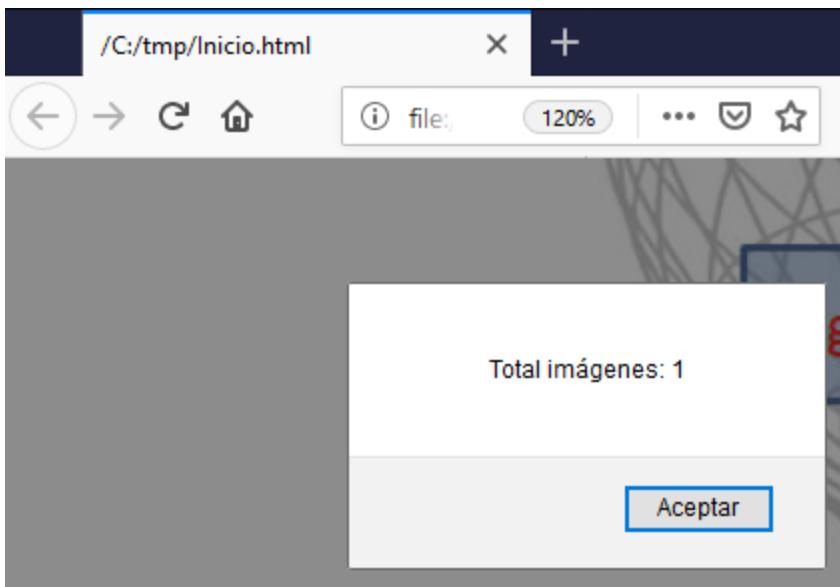


Ilustración 252: Cuenta el número de imágenes en el documento

## Describir los objetos de la página

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Describe los objetos de la página
function Atributos(){
    var cajatexto = document.getElementsByTagName("p")[0];
    alert("Total atributos del texto: " + cajatexto.attributes.length);
    alert("Nombre atributo 1: " + cajatexto.attributes[0].name);
    alert("Valor del atributo 1: " + cajatexto.attributes[0].value);
    alert("Nombre atributo 2: " + cajatexto.attributes[1].name);
    alert("Valor del atributo 2: " + cajatexto.attributes[1].value);
}
</script></head>
<body>
<p id="identifica" name="textual">Esto es una prueba</p>
<input type="button" onclick="Atributos()" value="¿Atributos del texto?">
</body></html>
```

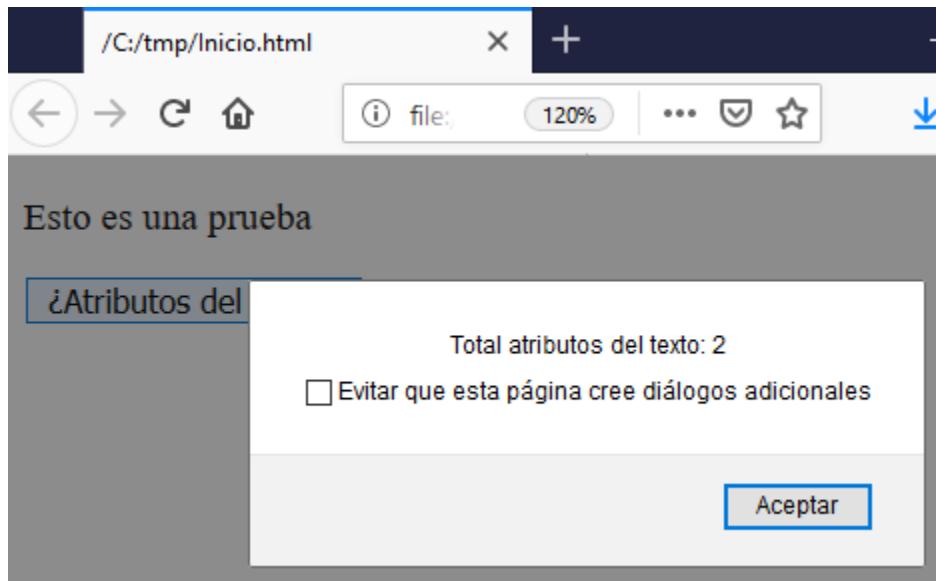


Ilustración 253: Cuenta el número de atributos que tenga un objeto

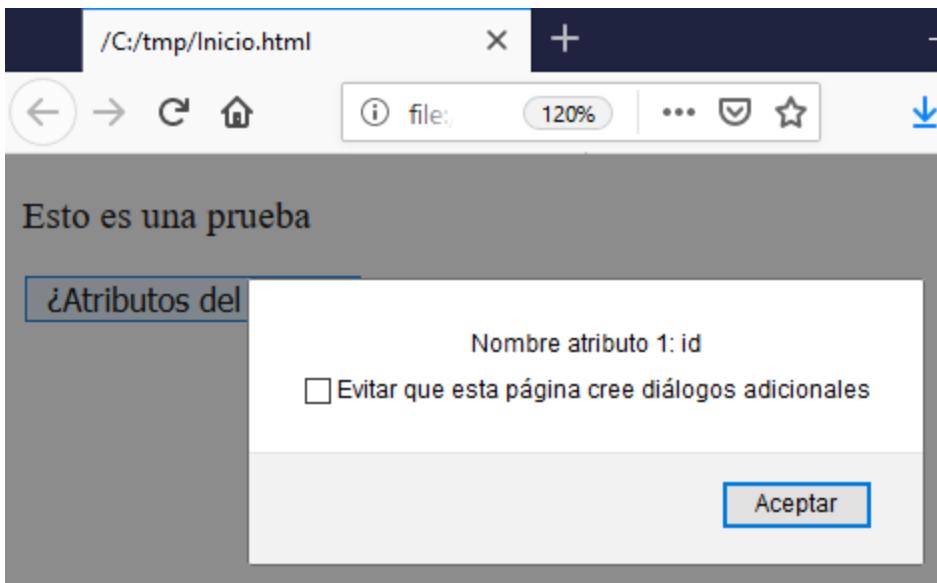


Ilustración 254: Trae la identificación de un objeto

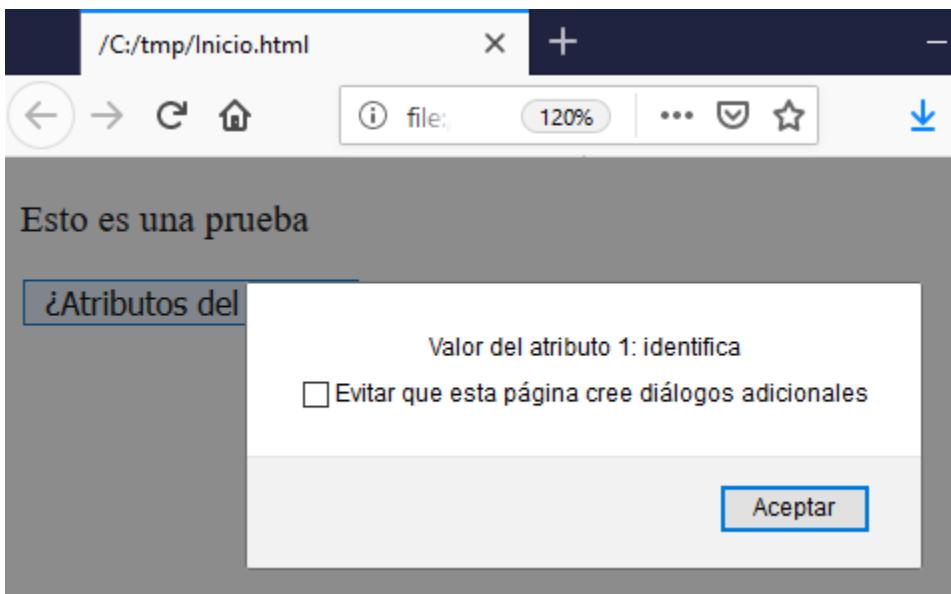


Ilustración 255: Trae el valor de la identificación de ese atributo

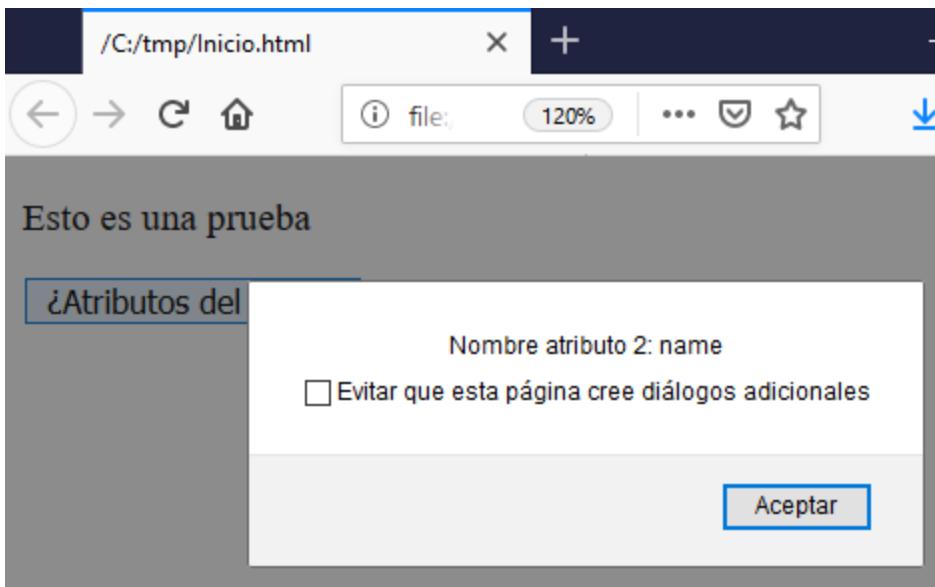


Ilustración 256: Trae otro atributo

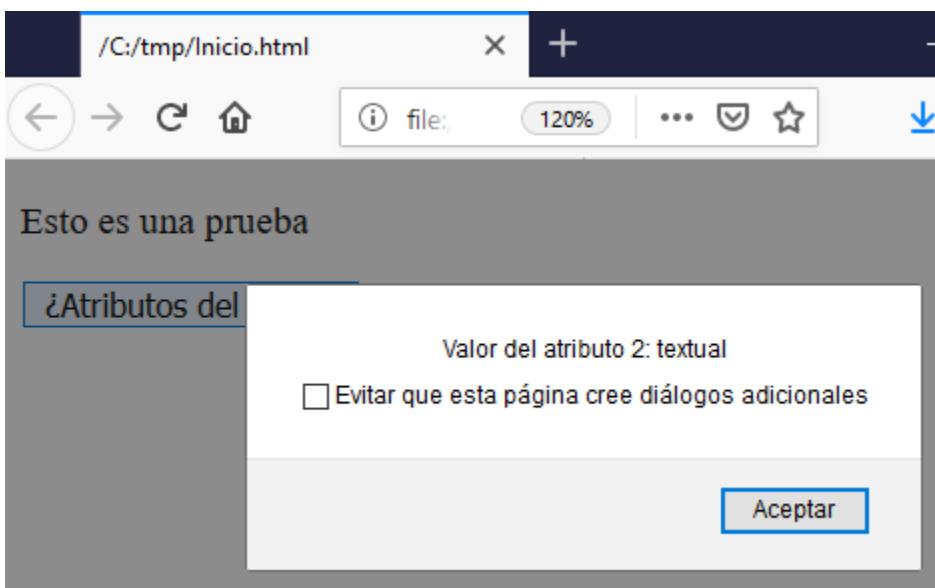


Ilustración 257: Muestra el valor de ese segundo atributo

## Abrir una ventana

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Abre una ventana secundaria con un ancho y largo determinados
function abrirventana() {
    window.open("http://darwin.50webs.com", "Investigación",
"width=300,height=200");
}
</script></head>
<body>
    <input type="button" value="Abrir ventana" onclick="abrirventana()"
/>
</body></html>
```

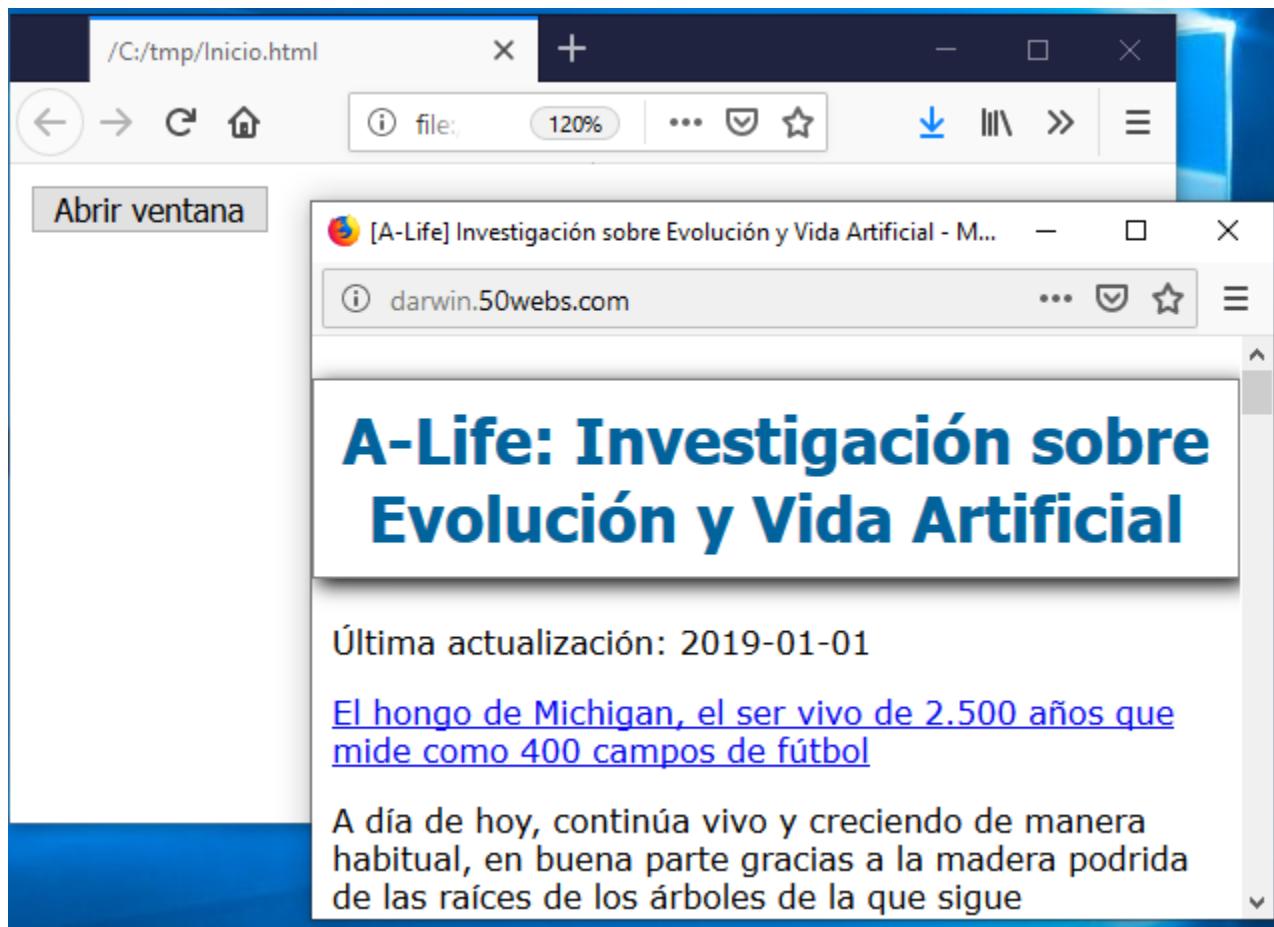


Ilustración 258: Abre una nueva ventana y apunta a una determinada dirección

## Abrir un conjunto de ventanas

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
//Abre determinadas ventanas apuntando a diferentes direcciones
var ventana=1;
function abrirventana() {
    switch(ventana) {
        case 1: direccion = "http://darwin.50webs.com"; break;
        case 2: direccion = "http://es.wikipedia.org"; break;
        case 3: direccion = "https://openlibra.com/es/book/redes-
neuronales-parte-1"; break;
    }
    ventana++;
    if (ventana==4) ventana=1;
    window.open(direccion, "_blank", "width=400,height=300");
}
</script></head>
<body>
    <input type="button" value="Abrir ventana" onclick="abrirventana()" />
</body></html>
```

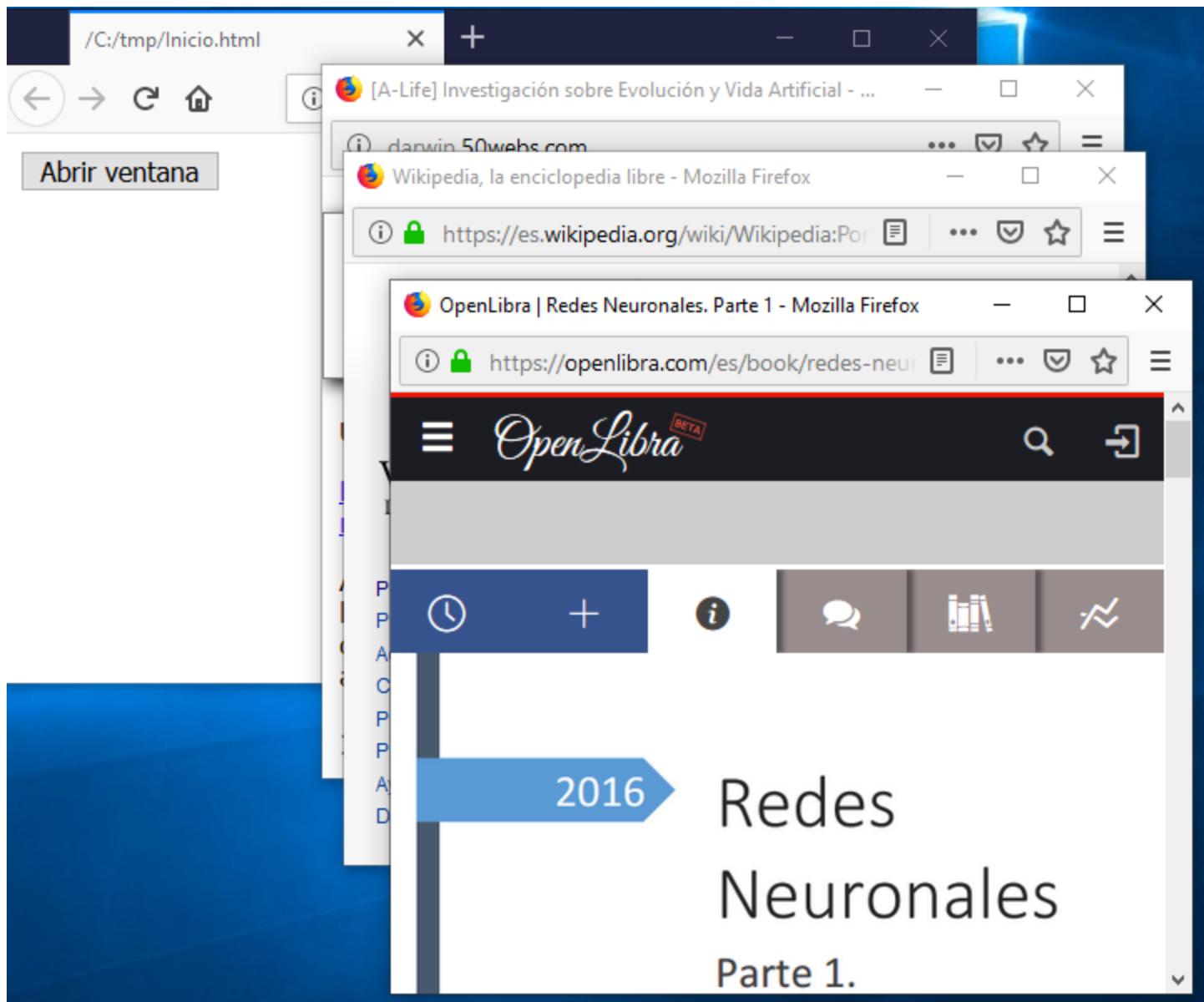


Ilustración 259: Abre varias ventanas, cada una hacia una dirección distinta

# Un formulario que ejecuta un algoritmo local al enviar la información

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
/*¿Cuántos números hay entre limitemin y limitemax que las dos últimas
cifras son múltiplo de 7?
Se muestra un formulario en el que el usuario digita los valores
limitemin y limitemax y el
programa contesta en el mismo formulario */
function calcula(){
    var cuenta = 0; var cadena = "";
    var limiteminimo =
parseInt(document.getElementById("limitemin").value);
    var limitemaximo =
parseInt(document.getElementById("limitemax").value);
    for (var numero=limiteminimo; numero<=limitemaximo; numero++)
        if ((numero%100)%7==0){
            cuenta++;
            cadena += numero + ", ";
        }
    document.getElementById("resultado").value = cuenta;
    document.getElementById("datos").value = cadena;
}
</script></head><body>
<form id="formulario" onsubmit="calcula(); return false;">
    <p>¿Cuántos números hay entre "Mínimo" y "Máximo" que las dos
últimas cifras son múltiplo de 7?</p>
    Mínimo: <input type="text" id="limitemin" /><br>
    Máximo: <input type="text" id="limitemax" /><br>
    <input type="submit" id="boton" value="envía" /><br>
</form>
Total: <input type="text" id="resultado" /><br>
Números: <input type="text" size="200" id="datos" />
</body></html>
```

The screenshot shows a web browser window with the URL `/C:/tmp/Inicio.html`. The page contains a form with the following fields:

- Mínimo:
- Máximo:
- 
- Total:
- Números:  
`1000, 1007, 1014, 1021, 1028, 1035, 1042, 1049, 1056, 1063, 1070, 1077, 1084`

Ilustración 260: Un formulario que ejecuta un algoritmo local al enviar la información

Una segunda versión del programa modificando el uso del “submit”

Directorio 048

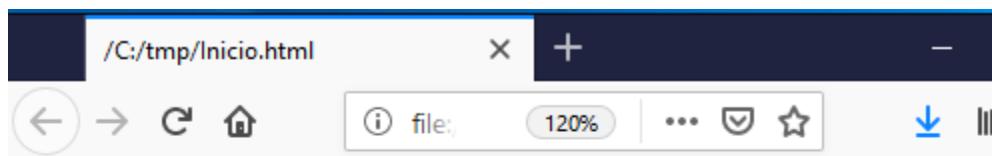
```
<!DOCTYPE HTML><html><head><script>
/*¿Cuántos números hay entre limitemin y limitemax que las dos últimas
cifras son múltiplo de 7?
Se muestra un formulario en el que el usuario digita los valores
limitemin y limitemax y el
programa contesta en el mismo formulario */
function calcula(){
    var cuenta = 0; var cadena = "";
    var limiteminimo =
parseInt(document.getElementById("limitemin").value);
    var limitemaximo =
parseInt(document.getElementById("limitemax").value);
    for (var numero=limiteminimo; numero<=limitemaximo; numero++)
        if ((numero%100)%7==0){
            cuenta++;
            cadena += numero + ", ";
        }
    document.getElementById("resultado").value = cuenta;
    document.getElementById("datos").value = cadena;
}
</script></head><body>
<form id="formulario">
    <p>¿Cuántos números hay entre "Mínimo" y "Máximo" que las dos
últimas cifras son múltiplo de 7?</p>
    Mínimo: <input type="text" id="limitemin" /><br>
```

```
Máximo: <input type="text" id="limitemax" /><br>
          <input type="button" value="Calcular valor"
        onclick='calcula()' />
      </form>
Total: <input type="text" id="resultado" /><br>
Números: <input type="text" size="200" id="datos" />
</body></html>
```

## Valida un correo con una expresión regular

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
/* Valida un correo con una expresión regular */
function validacorreo() {
    var correo = document.getElementById("direccion").value;
    var exprReg =
/^(([^\<^\>()[]\.\,\,;\:\s@\"]+(\.\[^<^\>()[]\.\,\,;\:\s@\""]+)*|(\\".+\\")*)@((\[[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-\-0-9]+\.)+[a-zA-
Z]{2,}))$/;
    var resultado = exprReg.test(correo);
    if (resultado)
        document.getElementById("mensaje").value = "La dirección de
correo electrónico es correcta";
    else
        document.getElementById("mensaje").value = "Es errónea la
dirección";
}
</script></head><body>
<form id="formulario" onsubmit="validacorreo(); return false;">
    <p>Ingrese dirección de correo electrónico</p>
    Correo: <input type="text" id="direccion" size="50" /><br>
    <input type="submit" id="boton" value="envía" /><br>
</form>
Resultado: <input type="text" size="50" id="mensaje" /><br>
</body></html>
```



Ingrese dirección de correo electrónico

Correo: probar

**envía**

Resultado: Es errónea la dirección

Ilustración 261: Valida el correo electrónico

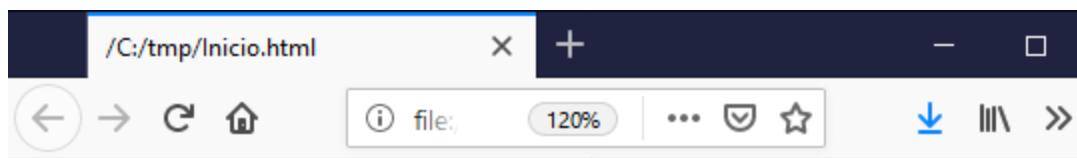
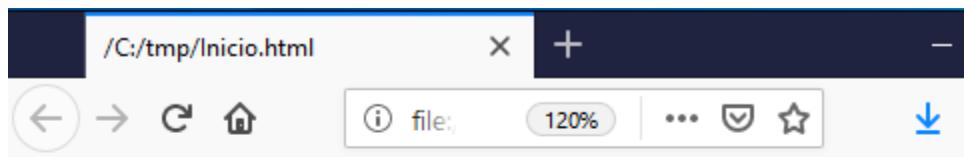


Ilustración 262: Valida el correo electrónico

## Valida una URL con expresiones regulares

Directorio 048

```
<!DOCTYPE HTML><html><head><script>
/* Valida una URL con expresiones regulares */
function validaURL() {
    var url = document.getElementById("direccion").value;
    var re = /^(ht|f)tp(s?)\:\/\/[0-9a-zA-Z]([-.\w]*[0-9a-zA-Z])*(:0-
9)*\*\/?)( [a-zA-Z0-9\-\.\?\,\'\|\|\+\&%\$#\_]*)?$/;
    var resultado = re.test(url);
    if (resultado)
        document.getElementById("mensaje").value = "La URL es correcta";
    else
        document.getElementById("mensaje").value = "Es errónea la URL";
}
</script></head><body>
<form id="formulario" onsubmit="validaURL(); return false;">
    <p>Ingrese una URL</p>
    URL: <input type="text" size="50" id="direccion" /><br>
    <input type="submit" id="boton" value="envía" /><br>
</form>
Mensaje: <input type="text" size="50" id="mensaje" /><br>
</body></html>
```



Ingrese una URL

URL:

Mensaje:

Ilustración 263: Valida la URL

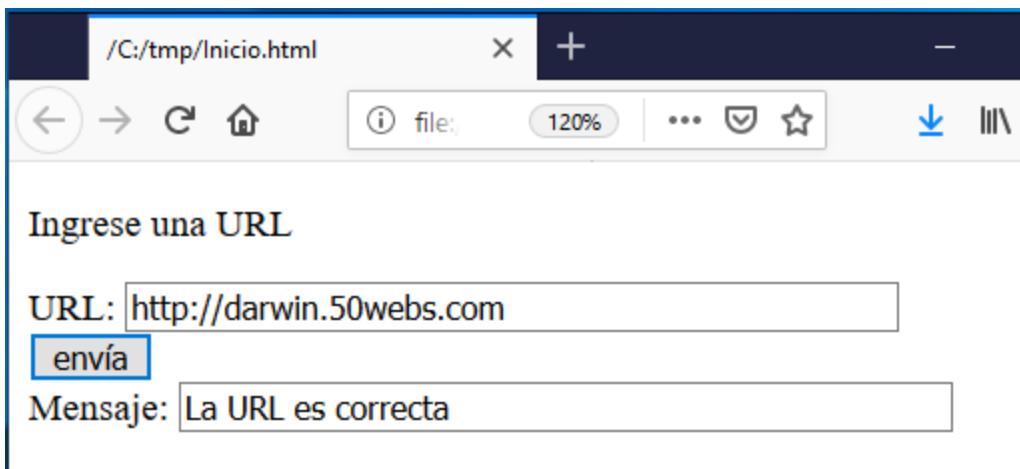


Ilustración 264: Valida la URL

## Gráficos

Con HTML5 se tiene los lienzos (canvas), zonas en las cuales se pueden dibujar líneas, óvalos, polígonos, etc. Con JavaScript se hacen los gráficos.

El primer paso es definir el lienzo (canvas) para dibujar y eso es con HTML5. Se debe especificar un “id” que será necesario para que JavaScript pueda trabajar con este.

Código básico:

*Directorio 049*

```
<!DOCTYPE HTML><html><head></head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
//Funciones para hacer gráficos

</script>
</body></html>
```

Si se quiere ver dónde está el lienzo, se añade un borde:

*Directorio 049*

```
<!DOCTYPE HTML><html><head></head>
<body>
<canvas id="milienzo" width="200" height="200" style="border:1px solid
#000000;"></canvas>
<script>
//Funciones para hacer gráficos

</script>
</body></html>
```

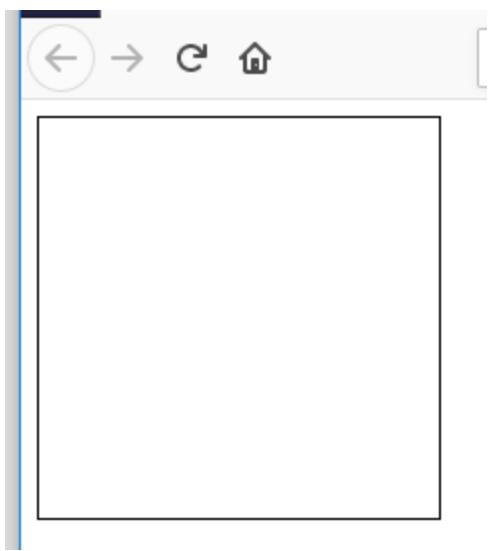


Ilustración 265:Lienzo (“canvas”) visible usando un borde

## Dibujar líneas

Usamos entonces instrucciones de JavaScript para hacer los gráficos. Se dibuja una línea.

Directorio 049

```
<!DOCTYPE HTML><html><head></head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
//Funciones para hacer gráficos

//Captura el lienzo
var lienzo = document.getElementById('milienzo');

//Captura el contexto (siempre es en 2d)
var contexto = lienzo.getContext('2d');

contexto.beginPath(); //Inicia el camino para dibujar
contexto.moveTo(50, 70); //Mueve el cursor a la posición X=50, Y=70
contexto.lineTo(140, 250); //Hace una línea entre (50,70) - (140,250)
contexto.stroke(); //Hace visible la línea

</script>
</body></html>
```



Ilustración 266: Línea dibujada

Se cambia el grueso de esa línea

```
<!DOCTYPE HTML><html><head></head>
<body>
<canvas id="miliendo" width="400" height="400"></canvas>
<script>
//Funciones para hacer gráficos

//Captura el lienzo
var lienzo = document.getElementById('miliendo');

//Captura el contexto (siempre es en 2d)
var contexto = lienzo.getContext('2d');

contexto.beginPath(); //Inicia el camino para dibujar
contexto.moveTo(50, 70); //Mueve el cursor a la posición X=50, Y=70
contexto.lineTo(140, 250); //Hace una línea entre (50,70) - (140,250)
contexto.lineWidth = 20; //Grosor de esa línea
contexto.stroke(); //Hace visible la línea

</script>
</body></html>
```



Ilustración 267: Línea gruesa

Se da un color determinado a esa línea

```

<!DOCTYPE HTML><html><head></head>
<body>
<canvas id="miliendo" width="400" height="400"></canvas>
<script>
//Funciones para hacer gráficos

//Captura el lienzo
var lienzo = document.getElementById('miliendo');

//Captura el contexto (siempre es en 2d)
var contexto = lienzo.getContext('2d');

contexto.beginPath(); //Inicia el camino para dibujar
contexto.moveTo(50, 70); //Mueve el cursor a la posición X=50, Y=70
contexto.lineTo(140, 250); //Hace una línea entre (50,70) - (140,250)
contexto.lineWidth = 20; //Grosor de esa línea

contexto.strokeStyle = '#8A2BE2'; //Color de la línea
contexto.stroke(); //Hace visible la línea

</script>
</body></html>

```



Ilustración 268: Color a la línea

El color puede ser expresado en RGB (Red, Green Blue). Conociendo los valores de esos colores primarios se pueden combinar.

```

<!DOCTYPE HTML><html><head></head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
//Funciones para hacer gráficos

//Captura el lienzo
var lienzo = document.getElementById('milienzo');

//Captura el contexto (siempre es en 2d)
var contexto = lienzo.getContext('2d');

contexto.beginPath(); //Inicia el camino para dibujar
contexto.moveTo(50, 70); //Mueve el cursor a la posición X=50, Y=70
contexto.lineTo(140, 250); //Hace una línea entre (50,70) - (140,250)
contexto.lineWidth = 10; //Grosor de esa línea
contexto.strokeStyle = rgbToHexadecimal(19, 163, 204); //Color de la línea
contexto.stroke(); //Hace visible la línea

//Convierte los números RGB (Red, Green, Blue) en su equivalente
hexadecimal de color
function rgbToHexadecimal(R,G,B){
    return "#" + Hexadec(R) + Hexadec(G) + Hexadec(B);
}

//Convierte a Hexadecimal
function Hexadec(num){
    num = parseInt(num, 10);
    if (isNaN(num)) return "00";
    num = Math.max(0, Math.min(num,255));
    return "0123456789ABCDEF".charAt((num-
num%16)/16)+"0123456789ABCDEF".charAt(num%16);
}
</script>
</body></html>

```



Ilustración 269: Línea con color modificando los parámetros RGB

## Estilos para finalizar las líneas

Directorio 049

```
<!DOCTYPE HTML><html><head></head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
//Funciones para hacer gráficos

//Captura el lienzo
var lienzo = document.getElementById('milienzo');

//Captura el contexto (siempre es en 2d)
var contexto = lienzo.getContext('2d');

contexto.beginPath(); //Inicia el camino para dibujar
contexto.lineWidth = 20; //Grosor de esa línea
contexto.strokeStyle = '#8A2BE2'; //Color de la línea
contexto.moveTo(50, 20); //Mueve el cursor a la posición X=50, Y=20
contexto.lineTo(50, 250); //Hace una línea entre (50,20) - (50,250)

//Estilo de inicio y fin de las líneas
contexto.lineCap = 'round'; //Otras opciones son: 'butt' y 'square'

contexto.stroke(); //Hace visible la línea
</script>
</body></html>
```



Ilustración 270: Las líneas finalizan redondeadas en ambos extremos usando "round"

Dibujar tres líneas. Requiere llamar tres veces la instrucción beginPath(). Dos terminaciones distintas.

Directorio 049

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
//Captura el lienzo
var lienzo = document.getElementById('milienzo');

//Captura el contexto (siempre es en 2d)
var contexto = lienzo.getContext('2d');

//Línea 1
contexto.beginPath(); //Inicia el camino para dibujar
contexto.lineWidth = 20; //Grosor de esa línea
contexto.strokeStyle = '#8A2BE2'; //Color de la línea
contexto.moveTo(50, 20); //Mueve el cursor a la posición X=50, Y=20
contexto.lineTo(50, 250); //Hace una línea entre (50,20) - (50,250)
contexto.lineCap = 'butt'; //Cierra extremos con butt
contexto.stroke(); //Hace visible la línea

//Línea 2 (misma longitud de Línea 1)
contexto.beginPath(); //Inicia el camino para dibujar
contexto.lineWidth = 20; //Grosor de esa línea
contexto.strokeStyle = '#8A2BE2'; //Color de la línea
contexto.moveTo(80, 20); //Mueve el cursor a la posición X=80, Y=20
contexto.lineTo(80, 250); //Hace una línea entre (80,20) - (80,250)
contexto.lineCap = 'square'; //Cierra extremos con square
```

```

contexto.stroke(); //Hace visible la linea

//Línea 3 (misma longitud de Línea 1)
contexto.beginPath(); //Inicia el camino para dibujar
contexto.lineWidth = 20; //Grosor de esa línea
contexto.strokeStyle = '#8A2BE2'; //Color de la línea
contexto.moveTo(110, 20); //Mueve el cursor a la posición X=110, Y=20
contexto.lineTo(110, 250); //Hace una línea entre (110,20) - (110,250)
contexto.lineCap = 'round'; //Cierra extremos con square
contexto.stroke(); //Hace visible la linea

</script></body></html>

```



Ilustración 271:Tres líneas con la misma longitud pero diferente cierre de extremo

Juntar dos líneas y que el punto de la unión tenga una característica (como ser redondeado)

*Directorio 049*

```

<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="400" height="400"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');

contexto.lineWidth = 28; //Ancho de la linea
contexto.strokeStyle = 'green'; //Color

//Genera dos líneas y las junta
contexto.beginPath();

```

```

contexto.moveTo(10,20);
contexto.lineTo(150, 90); //Línea 1
contexto.lineTo(270, 10); //Línea 2
contexto.lineJoin = 'round'; //miter, round, bevel
contexto.stroke();
</script>
</body></html>

```

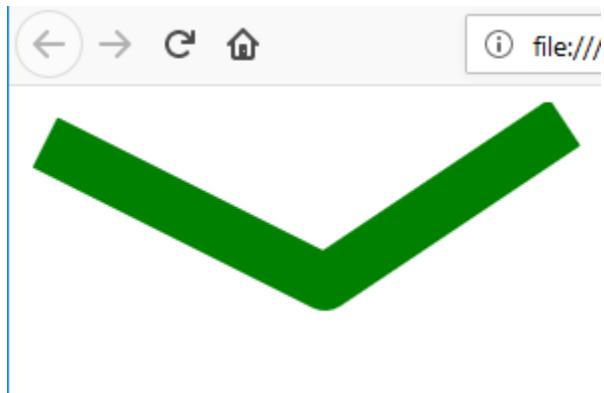


Ilustración 272: Dos líneas juntas con un empalme redondeado

Probando los otros estilos de juntar líneas

*Directorio 049*

```

<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="400" height="400"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');

contexto.lineWidth = 30; //Ancho de la línea
contexto.strokeStyle = 'green'; //Color

//Genera dos líneas y las junta
contexto.beginPath();
contexto.moveTo(10, 20);
contexto.lineTo(150, 90); //Línea 1
contexto.lineTo(270, 10); //Línea 2
contexto.lineJoin = 'round'; //miter, round, bevel
contexto.stroke();

//Genera dos líneas y las junta
contexto.beginPath();
contexto.moveTo(10, 80);
contexto.lineTo(150, 150); //Línea 1
contexto.lineTo(270, 70); //Línea 2
contexto.lineJoin = 'miter'; //miter, round, bevel
contexto.stroke();

```

```

//Genera dos líneas y las junta
contexto.beginPath();
contexto.moveTo(10, 140);
contexto.lineTo(150, 210); //Línea 1
contexto.lineTo(270, 130); //Línea 2
contexto.lineJoin = 'bevel'; //miter, round, bevel
contexto.stroke();
</script>
</body></html>

```

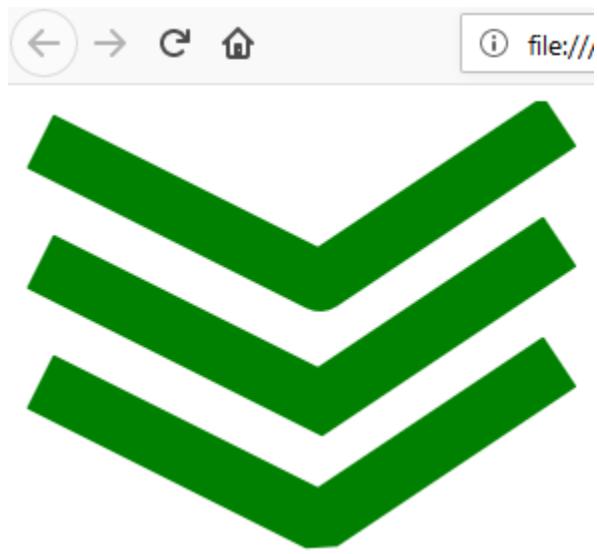


Ilustración 273: Diferentes formas de juntar dos líneas: "round", "miter", "bevel"

Con tres líneas se hace una figura y luego es cerrada (se hace una línea donde termina la tercera línea y donde empezó la primera línea).

*Directorio 049*

```

<!DOCTYPE HTML><html><head></head><body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
var lienzo = document.getElementById('milienzo');
var contexto = lienzo.getContext('2d');

contexto.lineWidth = 5; //Ancho de la línea
contexto.strokeStyle = 'green'; //Color

//Hace varias líneas continuas
contexto.beginPath();
contexto.moveTo(20, 30);
contexto.lineTo(250, 190); //Línea 1
contexto.lineTo(170, 50); //Línea 2

```

```

contexto.lineTo(280, 120); //Línea 3

//Hace una línea donde termina la línea 3 y empieza la línea 1
contexto.closePath();

contexto.stroke(); //Dibuja
</script>
</body></html>

```

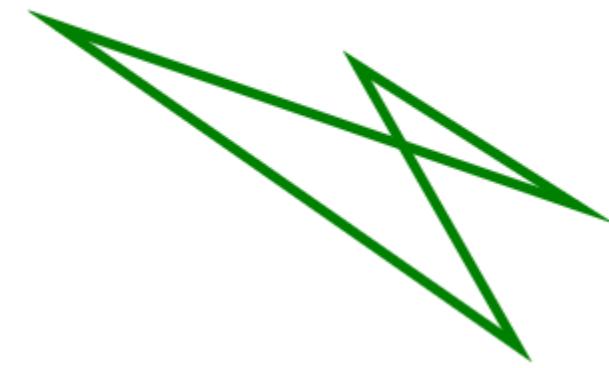
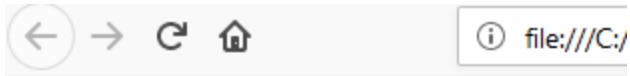


Ilustración 274: Una figura cerrada

Genera una figura cerrada y la rellena

Directorio 049

```

<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="400" height="400"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');

contexto.lineWidth = 5; //Ancho de la línea
contexto.strokeStyle = 'green'; //Color

//Hace una figura, cerrándola
contexto.beginPath();
contexto.moveTo(20,30);
contexto.lineTo(250, 190); //Línea 1
contexto.lineTo(170, 50); //Línea 2
contexto.closePath(); //Cierra la figura
contexto.stroke(); //Dibuja la figura

```

```

contexto.fillStyle = "red"; //Relleno rojo
contexto.fill(); //Rellena
</script>
</body></html>

```



Ilustración 275: Figura cerrada y rellena de un color

Dibujar varias líneas horizontales con un ciclo

*Directorio 049*

```

<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="400" height="400"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');

//Dibuja varias líneas horizontales
contexto.beginPath(); //Inicia el camino
for (var coordY=10; coordY<=500; coordY+=10){
    contexto.moveTo(5, coordY);
    contexto.lineTo(600, coordY);
}
contexto.stroke();
</script>
</body></html>

```

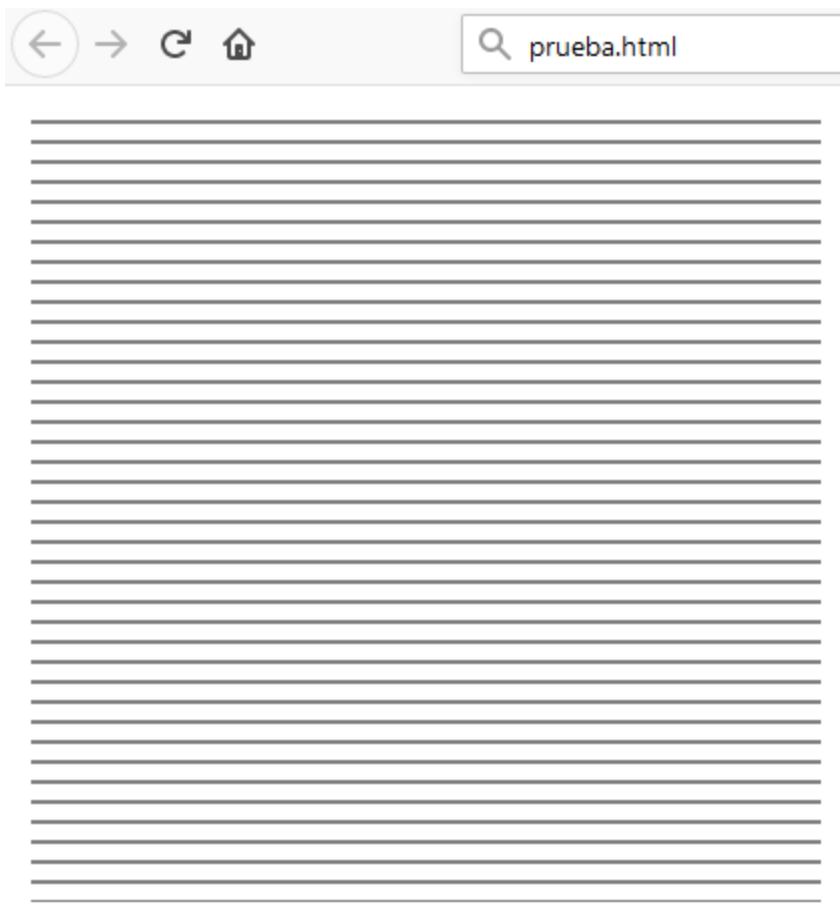


Ilustración 276: Líneas generadas por un ciclo

Dibujar varias líneas verticales con un ciclo

Directorio 049

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="milenzo" width="400" height="400"></canvas>
<script>
var lienzo = document.getElementById('milenzo');
var contexto = lienzo.getContext('2d');

//Dibuja varias líneas verticales
contexto.beginPath(); //Inicia el camino
for (var coordX=10; coordX<=500; coordX+=50){
    contexto.moveTo(coordX, 10);
    contexto.lineTo(coordX, 400);
}
contexto.stroke();
</script>
</body></html>
```

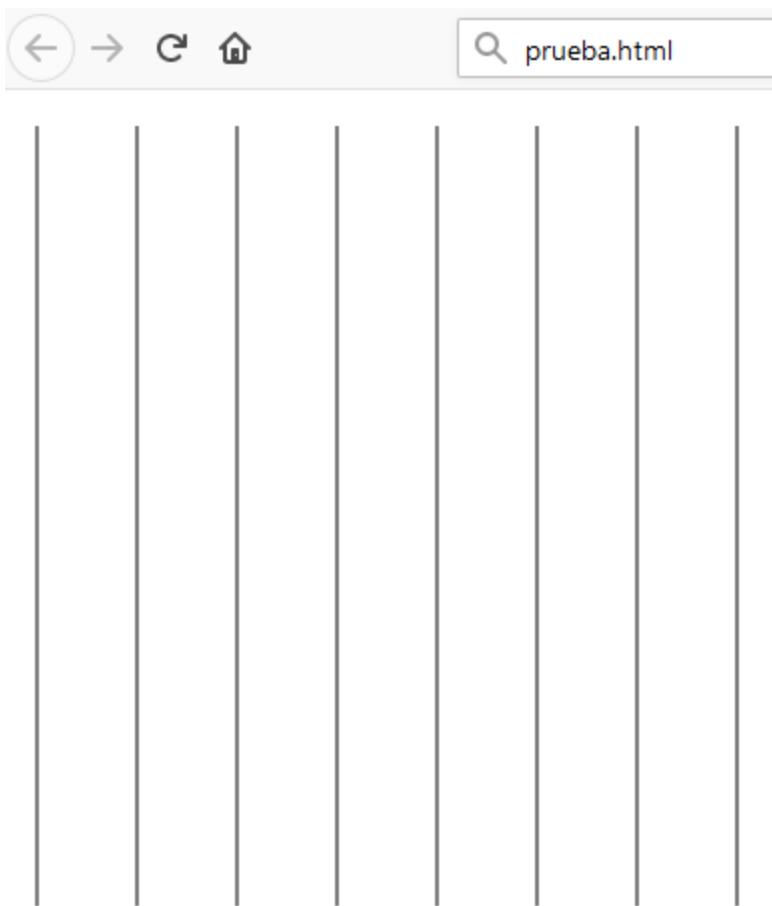


Ilustración 277: Líneas verticales dibujadas con un ciclo

### Ilusión de curva con varias líneas

Directorio 049

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="600" height="600"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');

contexto.beginPath(); //Inicia el camino

//Ilusión de curva con líneas
for (var mover=0; mover<=600; mover+=10){
    contexto.moveTo(0, mover);
    contexto.lineTo(mover, 600);
}
contexto.stroke();
</script>
</body></html>
```

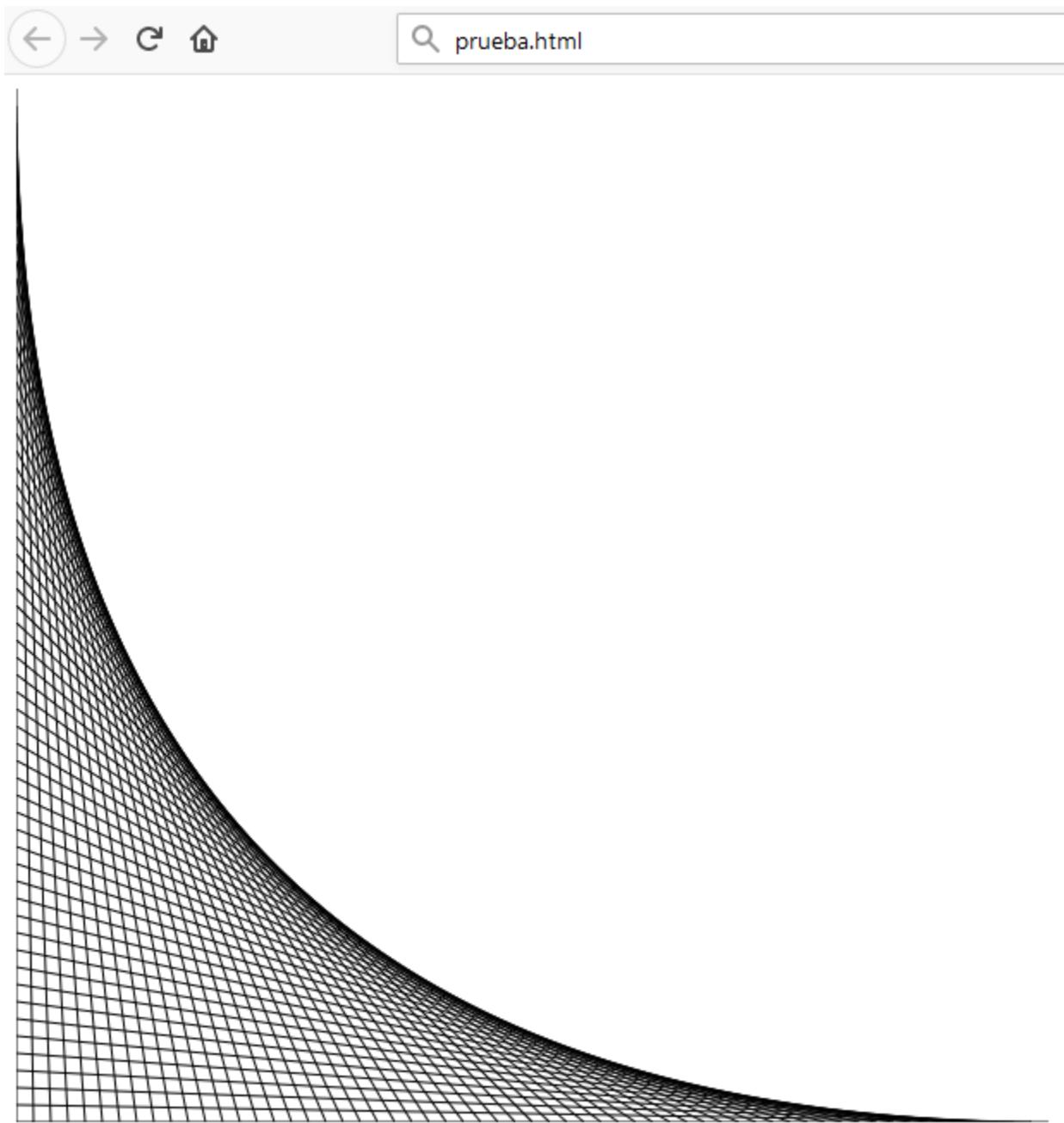


Ilustración 278: Ilusión de curva usando líneas rectas

“Curvas” con líneas

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="milenzo" width="600" height="600"></canvas>
<script>
var lienzo = document.getElementById('milenzo');
var contexto = lienzo.getContext('2d');

contexto.beginPath(); //Inicia el camino

//Curvas" usando líneas
for (var mover=0; mover<=600; mover+=10) {
    contexto.moveTo(0, mover);
    contexto.lineTo(mover, 600);

    contexto.moveTo(mover, 0);
    contexto.lineTo(600, mover);
}
contexto.stroke();
</script>
</body></html>
```

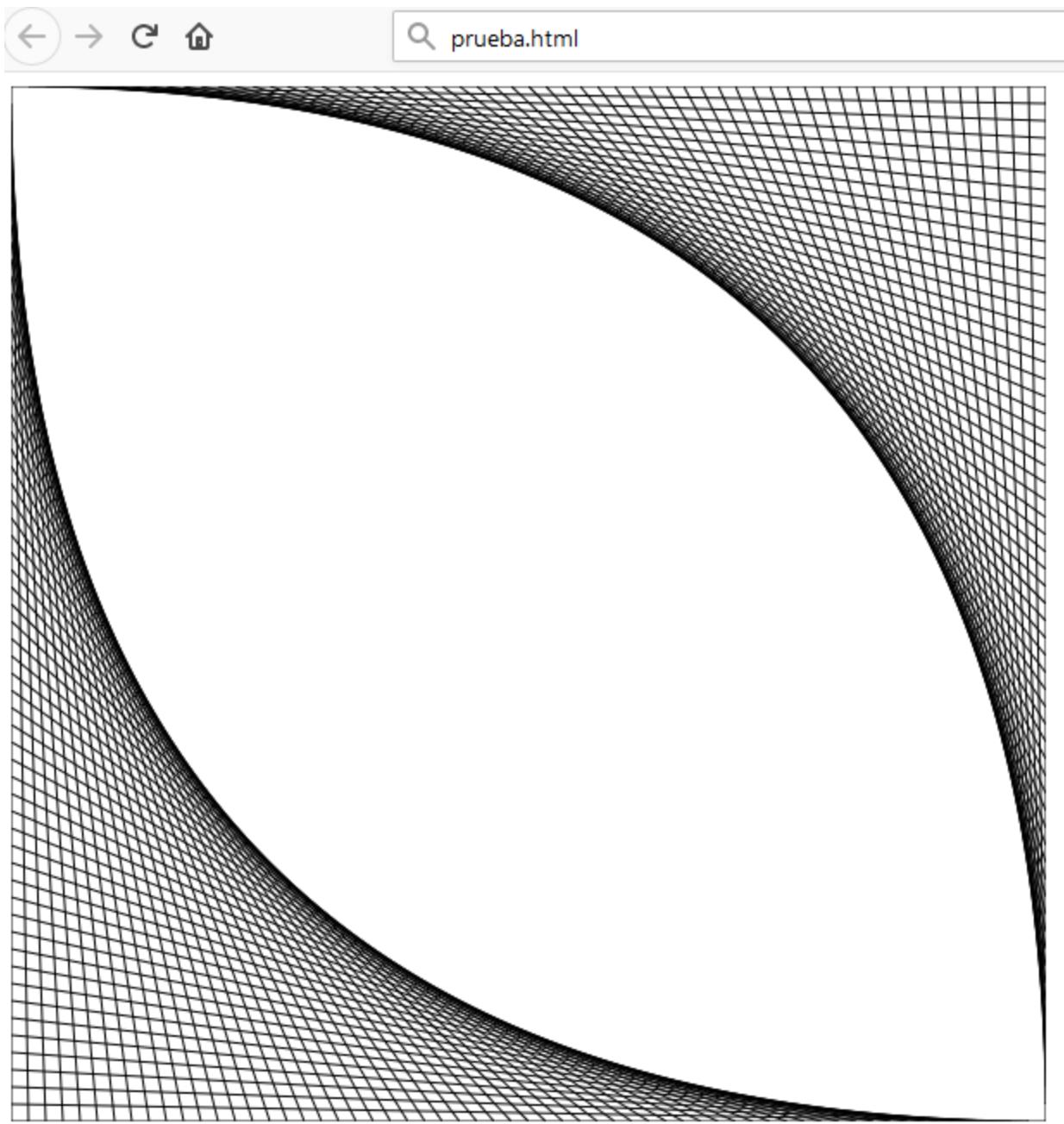


Ilustración 279: Ilusión de dos curvas usando líneas rectas

## Dibujar círculos

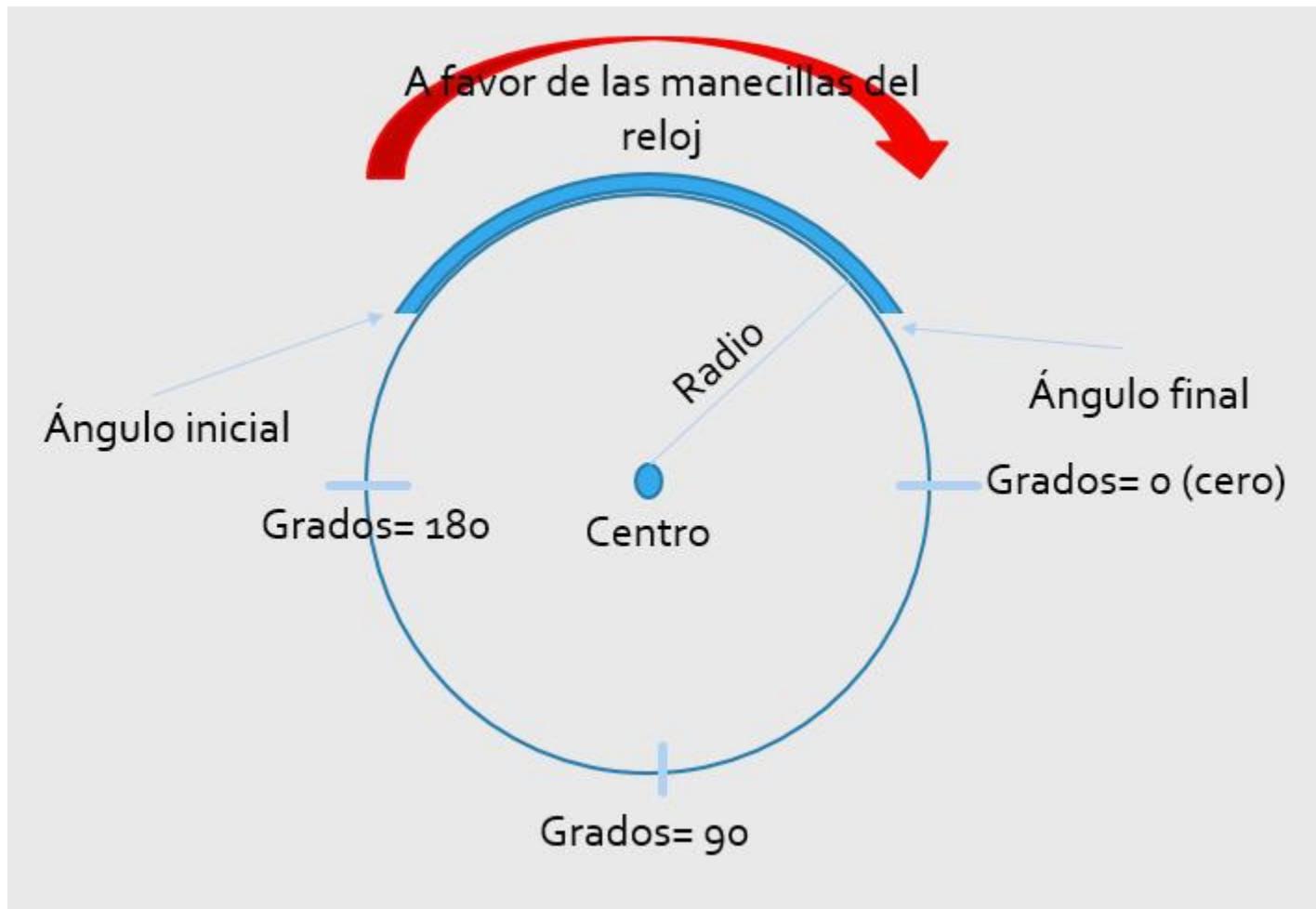


Ilustración 280: La manera en que se dibujará el círculo

Directorio 049

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="milienzo" width="300" height="300"></canvas>
<script>
var lienzo = document.getElementById('milienzo');
var contexto = lienzo.getContext('2d');

var posX = lienzo.width / 2; //Centro del canvas en X
var posY = lienzo.height / 2; //Centro del canvas en Y
var Radio = 100; //Radio del círculo
var AnguloInicio = 0 * Math.PI / 180; //Angulo inicial (0 grados) en radianes
var AnguloFinal = 360 * Math.PI / 180; //Angulo final (360 grados) en radianes
var DibujaContraReloj = false;

contexto.beginPath();
```

```
contexto.arc(posX, posY, Radio, AnguloInicio, AnguloFinal,  
DibujaContraReloj);  
contexto.lineWidth = 10; //Ancho de la línea  
contexto.strokeStyle = 'green'; //Color  
contexto.stroke();  
</script></body></html>
```

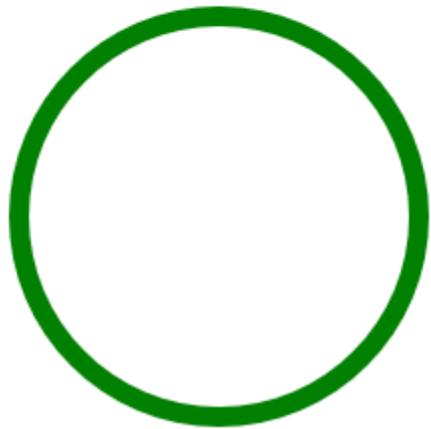
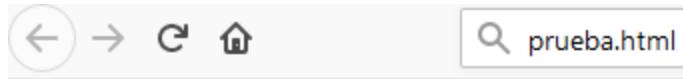


Ilustración 281: Círculo

## Dibujar curvas

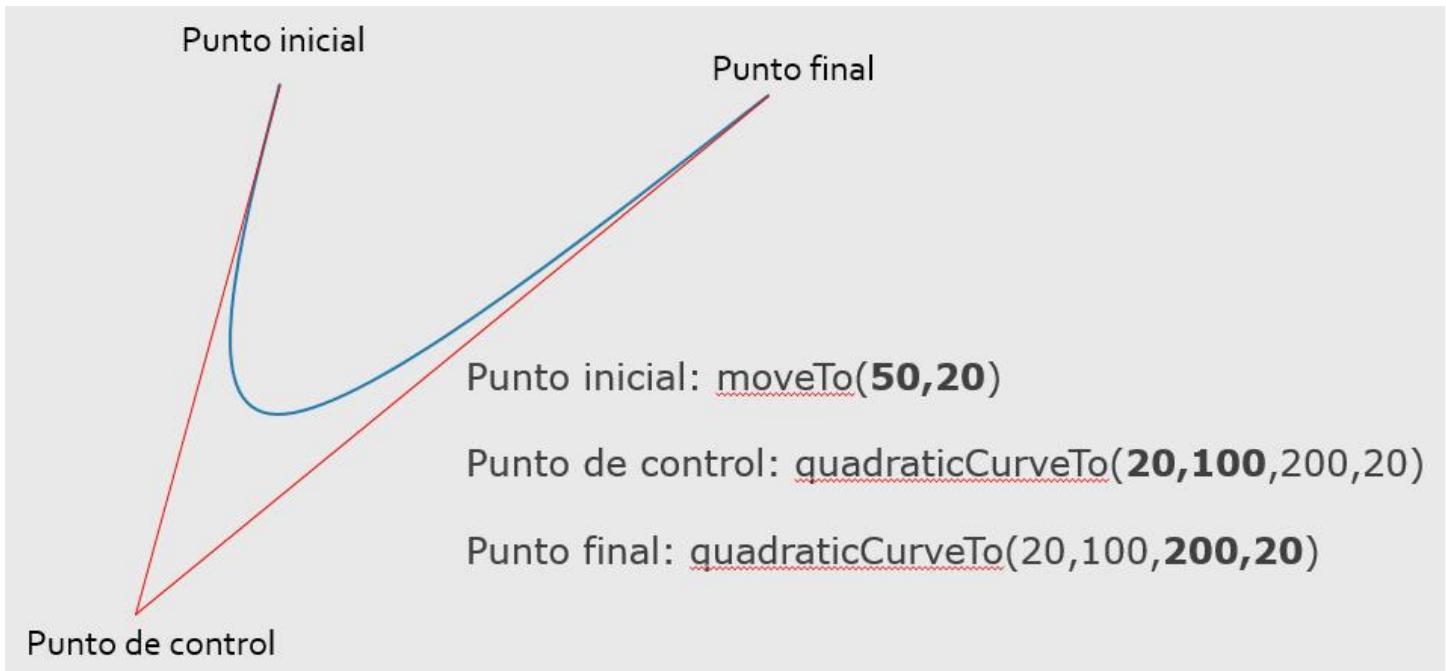


Ilustración 282: Forma en que se dibujarán las curvas

Directorio 049

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="300" height="300"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');

contexto.beginPath();

contexto.moveTo(50,20); //Punto inicial
contexto.quadraticCurveTo(20,100,200,20); //Punto de control y punto final

contexto.lineWidth = 5; //Ancho de la línea
contexto.strokeStyle = 'green'; //Color
contexto.stroke();
</script></body></html>
```

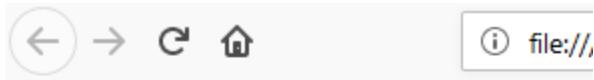
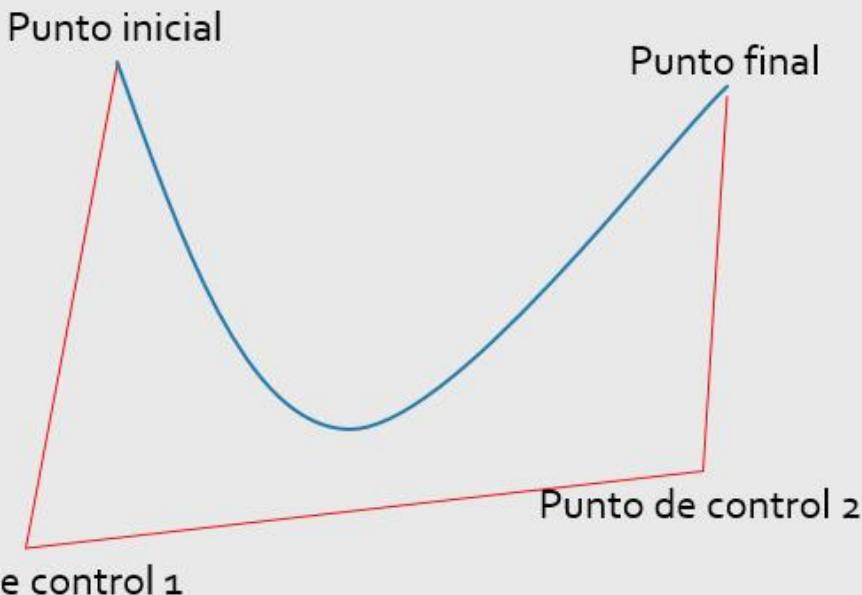


Ilustración 283: Curva

## Curva de Bézier



Punto inicial: `moveTo(80,20)`

Punto de control 1: `bezierCurveTo(60, 100, 200, 90, 240, 25)`

Punto de control 2: `bezierCurveTo(60, 100, 200, 90, 240, 25)`

Punto final: `bezierCurveTo(60, 100, 200, 90, 240, 25)`

Directorio 049

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="300" height="300"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');

contexto.beginPath();

contexto.moveTo(80,20); //Punto inicial
contexto.bezierCurveTo(60, 100, 200, 90, 240, 25);

contexto.lineWidth = 5; //Ancho de la linea
contexto.strokeStyle = 'green'; //Color
contexto.stroke();
</script>
</body></html>
```



Ilustración 284: Curva Bézier

## Combinando

[Directorio 049](#)

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="300" height="300"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');
contexto.lineWidth = 5; //Ancho de la linea

IniciaElementoGrafico(contexto, 'green', 10, 20);
contexto.bezierCurveTo(30, 280, 120, 5, 290, 25);
contexto.stroke();

IniciaElementoGrafico(contexto, 'blue', 290, 25);
contexto.lineTo(150, 90);
contexto.stroke();

IniciaElementoGrafico(contexto, 'black', 150, 90);
contexto.quadraticCurveTo(320, 250, 200, 20);
contexto.stroke();

//Esta función ahorra tener que repetir instrucciones cada vez
//que se quiera dibujar un nuevo objeto
function IniciaElementoGrafico(contexto, quecolor, posX, posY){
    contexto.beginPath();
    contexto.strokeStyle = quecolor;
    contexto.moveTo(posX, posY);
}
</script>
</body></html>
```

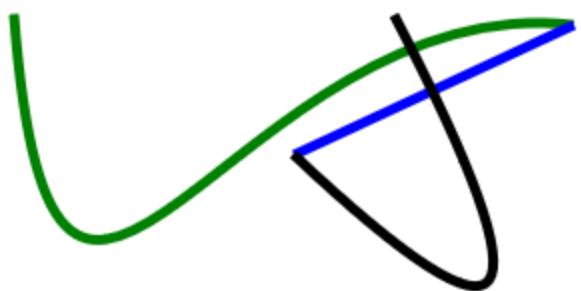


Ilustración 285: Combinando curvas

## Rectángulo y gradiente de color

Se puede tener un relleno de color que vaya cambiando de oscuro a más claro, o de un color a otro.

Directorio 049

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="400" height="400"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');

//Hace un rectángulo
contexto.rect(0, 0, lienzo.width, lienzo.height);

//Gradiente de cambio de color
var gradiente = contexto.createLinearGradient(0, 0, lienzo.width,
lienzo.height);
gradiente.addColorStop(0, '#000000'); //Punto inicial de primer color 0%
gradiente.addColorStop(1, '#FFFFFF'); //Punto final de último color 100%
contexto.fillStyle = gradiente;
contexto.fill(); //Rellena
</script>
</body></html>
```

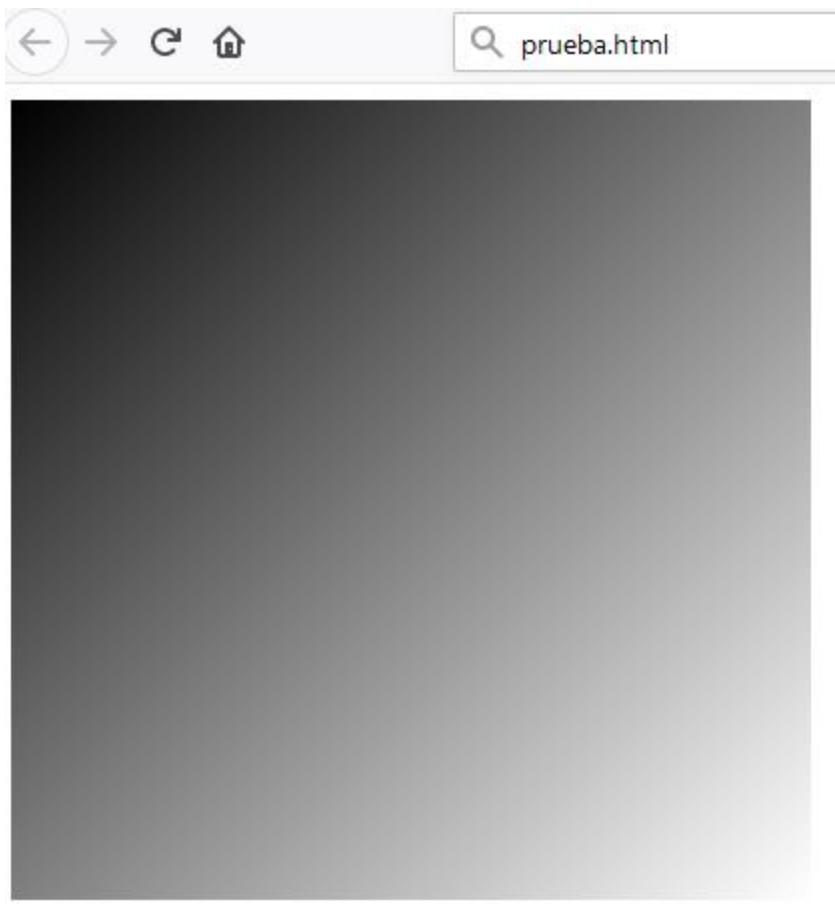


Ilustración 286: Variación del color

Variando en forma oblicua con varios colores

Directorio 049

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
var lienzo = document.getElementById('milienzo');
var contexto = lienzo.getContext('2d');

//Hace un rectángulo
contexto.rect(0, 0, lienzo.width, lienzo.height);

/* Crea un gradiente de color oblicuo: una línea imaginaria del punto
superior izquierdo al punto inferior derecho. */
var gradiente = contexto.createLinearGradient(0, 0, lienzo.width,
lienzo.height);

//Diferentes colores
gradiente.addColorStop(0, '#ABCDEF'); //Punto inicial de primer color
gradiente.addColorStop(0.2, '#BCD123');
```

```

gradiente.addColorStop(0.4, '#571ACB');
gradiente.addColorStop(0.6, '#C1B2D3');
gradiente.addColorStop(0.8, '#F1C5C2');
gradiente.addColorStop(1, '#012345'); //Punto final de último color
contexto.fillStyle = gradiente;
contexto.fill(); //Rellena
</script>
</body></html>

```

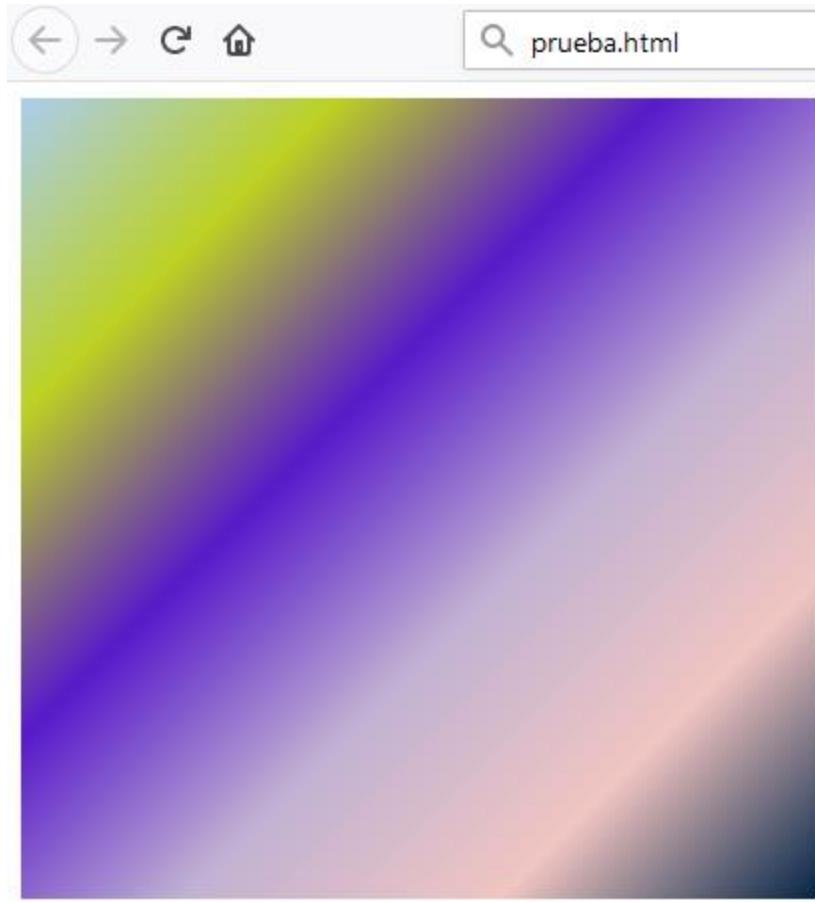


Ilustración 287: Variación de varios colores en forma oblicua

Variando en forma vertical

Directorio 049

```

<!DOCTYPE HTML><html><head></head><body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
var lienzo = document.getElementById('milienzo');
var contexto = lienzo.getContext('2d');

//Hace un rectángulo
contexto.rect(0, 0, lienzo.width, lienzo.height);

```

```

/* Crea un gradiente de color de variación vertical */
var gradiente = contexto.createLinearGradient(0, lienzo.height/2,
lienzo.width, lienzo.height/2);

//Diferentes colores
gradiente.addColorStop(0, '#ABCDEF'); //Punto inicial de primer color
gradiente.addColorStop(0.2, '#BCD123');
gradiente.addColorStop(0.4, '#571ACB');
gradiente.addColorStop(0.6, '#C1B2D3');
gradiente.addColorStop(0.8, '#F1C5C2');
gradiente.addColorStop(1, '#012345'); //Punto final de último color
contexto.fillStyle = gradiente;
contexto.fill(); //Rellena
</script>
</body></html>

```

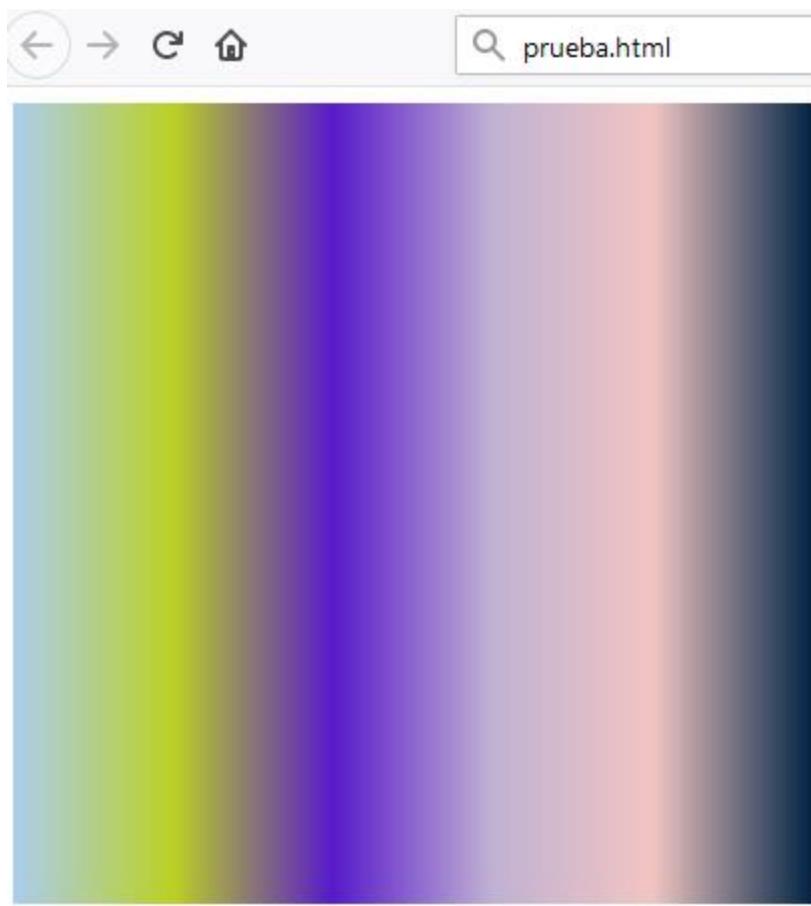


Ilustración 288: Variación de color en forma vertical

Variando en forma horizontal

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="milenzo" width="400" height="400"></canvas>
<script>
var lienzo = document.getElementById('milenzo');
var contexto = lienzo.getContext('2d');

//Hace un rectángulo
contexto.rect(0, 0, lienzo.width, lienzo.height);

/* Crea un gradiente de color de variación horizontal */
var gradiente = contexto.createLinearGradient(lienzo.width/2, 0,
lienzo.width/2, lienzo.height);

//Diferentes colores
gradiente.addColorStop(0, '#ABCDEF'); //Punto inicial de primer color
gradiente.addColorStop(0.2, '#BCD123');
gradiente.addColorStop(0.4, '#571ACB');
gradiente.addColorStop(0.6, '#C1B2D3');
gradiente.addColorStop(0.8, '#F1C5C2');
gradiente.addColorStop(1, '#012345'); //Punto final de último color
contexto.fillStyle = gradiente;
contexto.fill(); //Rellena
</script>
</body></html>
```

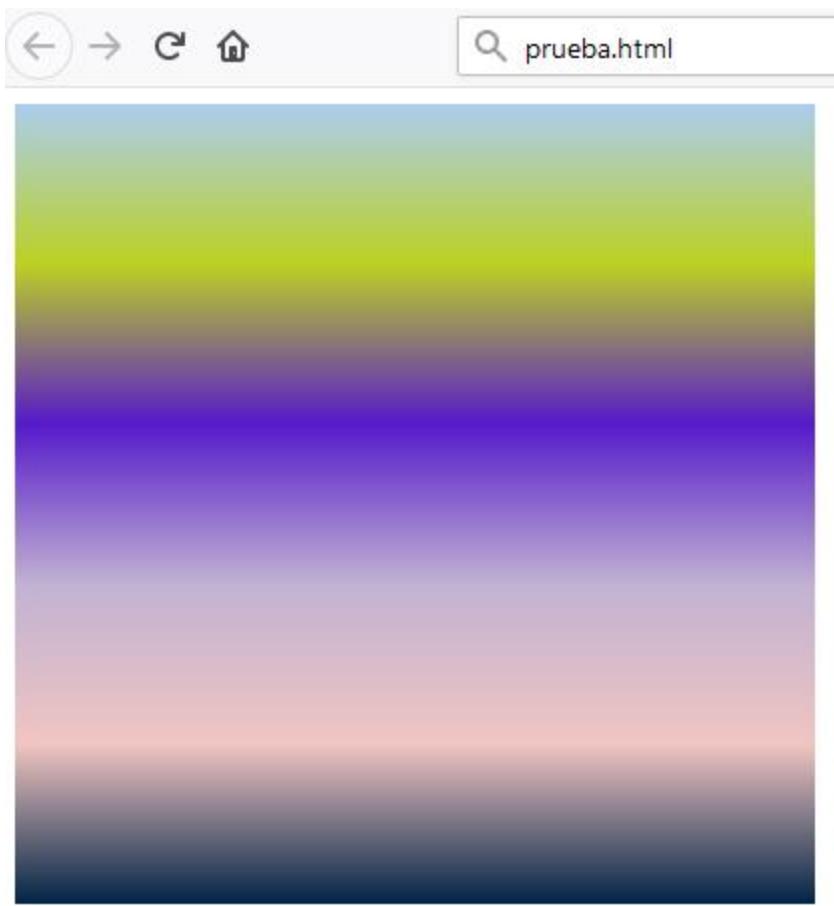


Ilustración 289: Variación de color horizontal

#### Variando en círculos

Directorio 049

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="400" height="400"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');

//Hace un rectángulo
contexto.rect(0, 0, lienzo.width, lienzo.height);

/* Crea un gradiente radial que se define como dos círculos imaginarios,
el que
    inicia y el que termina. (posX1, posY1, radio1, posX2, posY2, radio2 */
var gradiente = contexto.createRadialGradient(200, 200, 10, 200, 200,
200);

//Diferentes colores
gradiente.addColorStop(0, '#ABCDEF'); //Punto inicial de primer color
```

```
gradiente.addColorStop(0.2, '#BCD123');
gradiente.addColorStop(0.4, '#571ACB');
gradiente.addColorStop(0.6, '#C1B2D3');
gradiente.addColorStop(0.8, '#F1C5C2');
gradiente.addColorStop(1, '#012345'); //Punto final de último color
contexto.fillStyle = gradiente;
contexto.fill(); //Rellena
</script>
</body></html>
```

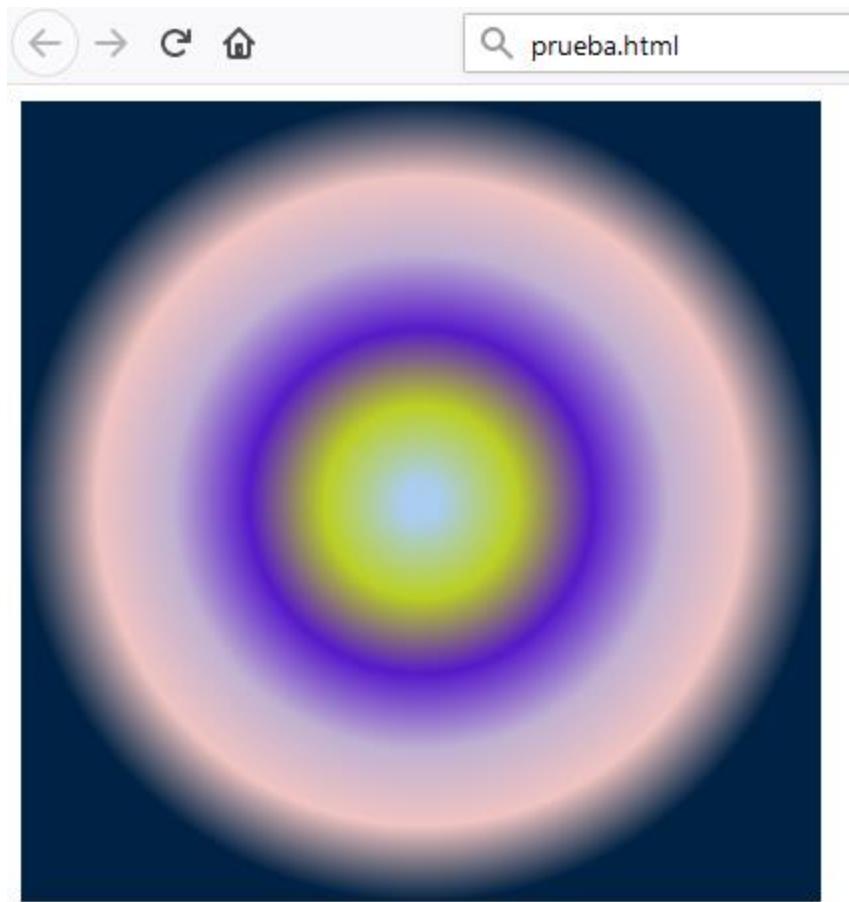


Ilustración 290: Variación en círculos

# Trabajando con imágenes

Directorio 049

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="400" height="400"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');

contexto.rect(0, 0, lienzo.width, lienzo.height); //Un rectángulo

imagen = new Image();
imagen.src = 'logo.jpg'; //La imagen está en un archivo .jpg

//Si la imagen está cargada
imagen.onload = function(){
    var patron = contexto.createPattern(imagen, 'repeat');
    contexto.fillStyle = patron;
    contexto.fill(); //Rellena
}

/*
Probemos con esto también:
var patron = contexto.createPattern(imagen, 'repeat-x');
var patron = contexto.createPattern(imagen, 'repeat-y');
var patron = contexto.createPattern(imagen, 'no-repeat'); */
</script>
</body></html>
```



Ilustración 291: La imagen rellena el rectángulo repitiéndose

Muestra la imagen, se puede variar la posición en el lienzo y su tamaño en ancho y alto

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="lienzo" width="400" height="400"></canvas>
<script>
var lienzo = document.getElementById('lienzo');
var contexto = lienzo.getContext('2d');

contexto.rect(0, 0, lienzo.width, lienzo.height); //Un rectángulo

imagen = new Image();
imagen.src = 'logo.jpg'; //La imagen está en un archivo .jpg

//Si la imagen está cargada
imagen.onload = function(){
    //Muestra la imagen, cambiando posición y tamaño de esta
    //posX, posY, ancho y alto
    contexto.drawImage(imagen, 50, 50, 300, 30);
}
</script>
</body></html>
```

## Convertir imagen a escala de grises

Directorio 049

```
<!DOCTYPE HTML><html><head></head>
<body><canvas id="miLienzo" width="595" height="420"></canvas>
<script>
var Imagen = new Image(); //Crea un objeto imagen
Imagen.src = 'Sally.jpg'; //donde se encuentra la imagen

Imagen.onload = function(){
    DibujarImagen(this);
}; //Cuando se cargue el objeto imagen

//Escala de grises
function DibujarImagen(Imagen) {
    var lienzo = document.getElementById('miLienzo');
    var contexto = lienzo.getContext('2d');
    contexto.drawImage(Imagen, 0, 0); //Dibuja la imagen original en 0,0

    //Trae los datos de esa imagen
    var datoImagen = contexto.getImageData(0, 0, Imagen.width,
Imagen.height);
    var Datos = datoImagen.data;

    //Se va de pixel en pixel y lo convierte a escala de grises
    for(var i = 0; i < Datos.length; i += 4) {
        var grisaceo = 0.34 * Datos[i] + 0.5 * Datos[i+1] + 0.16 *
Datos[i+2];
        Datos[i] = grisaceo;
        Datos[i + 1] = grisaceo;
        Datos[i + 2] = grisaceo;
    }
    contexto.putImageData(datoImagen, 0, 0); //Pone la imagen convertida
sobre la imagen original
}
</script>
</body></html>
```



Ilustración 292: Imagen original

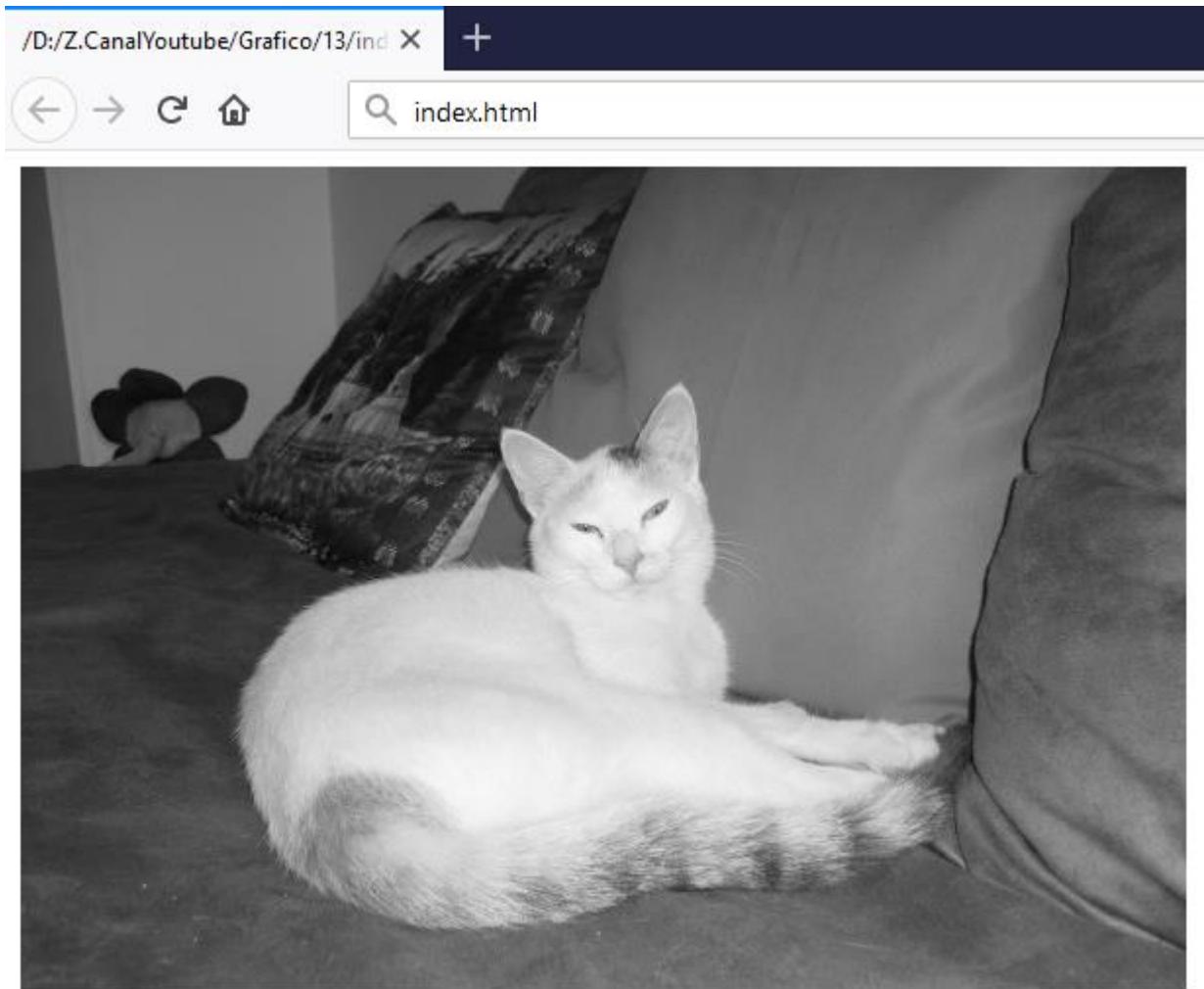


Ilustración 293: Imagen en escala de grises

## Dibujar letras

Se pueden imprimir textos como imágenes.

Directorio 049

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="700" height="400"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');

//Tipo de letra, tamaño, fuente
contexto.font = '60pt Courier';

//Texto a imprimir, posición X, posición Y
contexto.fillText('Esta es una prueba', 0, 150);
</script>
</body></html>
```



Esta es una pru

Ilustración 294: Dibuja un texto

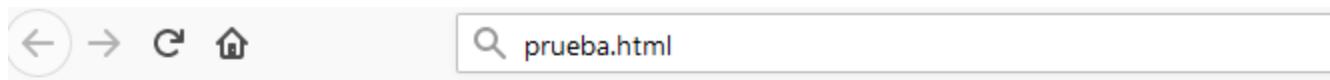
Directorio 049

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="700" height="400"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');

//Tipo de letra, tamaño, fuente
contexto.font = 'bold 45pt Verdana';

//Color que tendrá el texto
contexto.fillStyle = 'blue';
```

```
//Texto a imprimir, posición X, posición Y
contexto.fillText('Esta es una prueba', 10, 100);
</script>
</body></html>
```



# Esta es una prueba

Ilustración 295: Dibuja un texto con relleno de color

Directorio 049

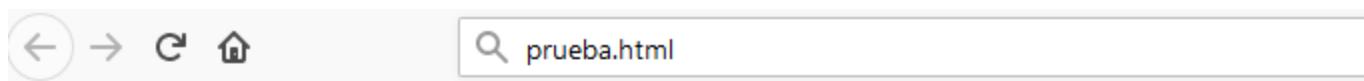
```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="700" height="400"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');

//Tipo de letra, tamaño, fuente
contexto.font = '50pt Verdana';

//Ancho de la línea
contexto.lineWidth = 1;

//Color de la línea
contexto.strokeStyle = 'orange';

//Texto a imprimir, posición X, posición Y
contexto.strokeText('Esta es una prueba', 10, 100);
</script>
</body></html>
```



# Esta es una prueba

Ilustración 296: Texto dibujado, se muestra el borde de cada letra

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="700" height="400"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');

//Dibuja una línea para hacer referencia
contexto.moveTo(350, 0);
contexto.lineTo(350, 350);
contexto.stroke();

//Tipo de letra, tamaño, fuente
contexto.font = 'bold 45pt Verdana';

//Alineación del texto: left, center, right, start, end
contexto.textAlign = 'end';

//Texto a imprimir, posición X, posición Y
contexto.fillText('Alinear', 350, 100);
</script>
</body></html>
```

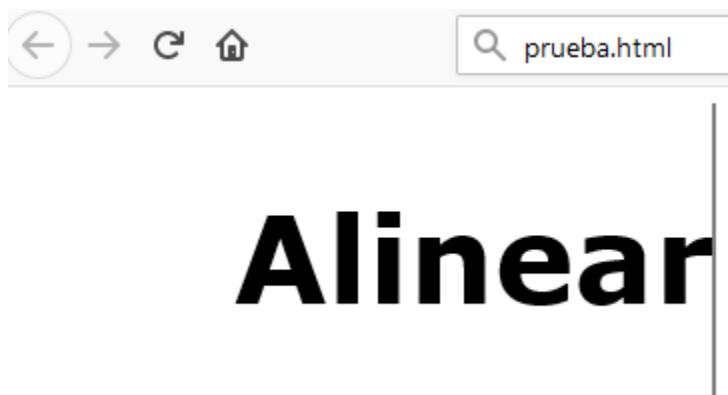


Ilustración 297: Dibuja un texto alineado al final

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="700" height="400"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');

//Tipo de letra, tamaño, fuente
contexto.font = '80pt Courier';

//Texto a imprimir, posición X, posición Y
var texto = "Prueba";
contexto.fillText(texto, 10, 100);
```

```
//Tamaño del texto
var tamano = contexto.measureText(texto);
var ancho = tamano.width;
alert(ancho);
</script>
</body></html>
```

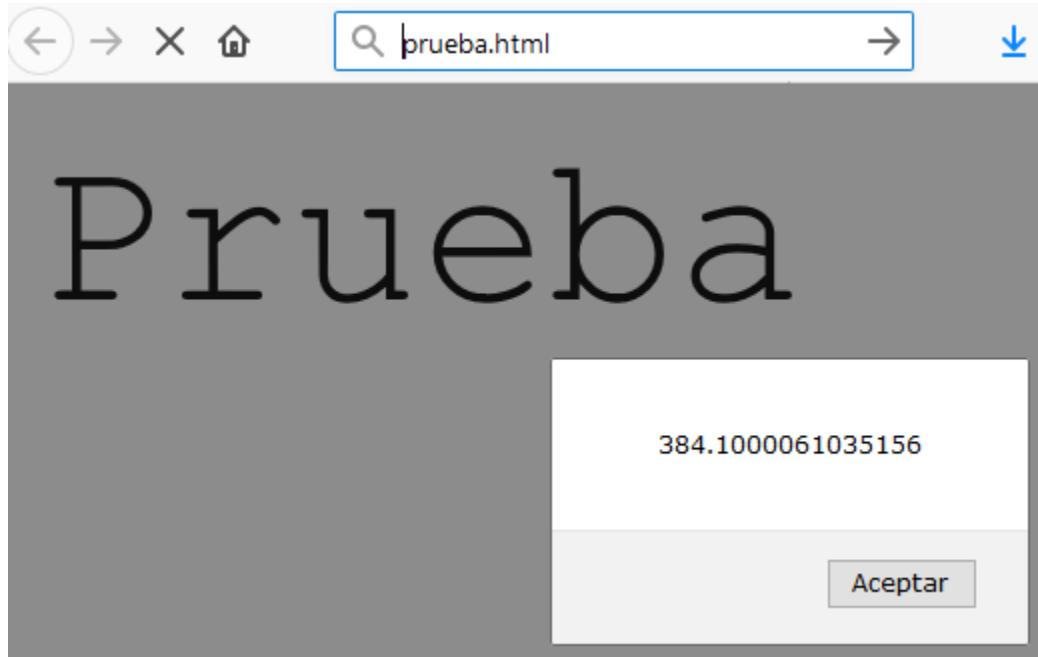


Ilustración 298: Muestra el ancho del texto dibujado

## Sombras

Directorio 049

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="700" height="400"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');

contexto.rect(50, 50, 200, 200);
contexto.fillStyle = 'green';

contexto.shadowColor = 'blue'; //Color de la sombra
contexto.shadowBlur = 50; //Definición de la sombra
contexto.shadowOffsetX = 40; //Corrimiento sombra en X
contexto.shadowOffsetY = 40; //Corrimiento sombra en Y

contexto.fill();
</script>
</body></html>
```

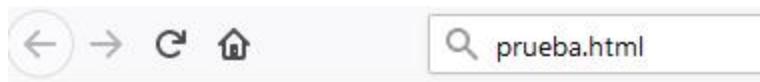


Ilustración 299: Sombra

## Semitransparente

Directorio 049

```
<!DOCTYPE HTML><html><head></head><body>
<canvas id="miliendo" width="700" height="400"></canvas>
<script>
var lienzo = document.getElementById('miliendo');
var contexto = lienzo.getContext('2d');

//Cuadrado 1
contexto.beginPath();
contexto.rect(10, 10, 200, 200);
contexto.fillStyle = 'blue';
contexto.fill();

//Cuadrado 2... semitransparente
contexto.globalAlpha = 0.4;
contexto.beginPath();
contexto.rect(100, 100, 200, 200);
contexto.fillStyle = 'gray';
contexto.fill();
</script>
</body></html>
```

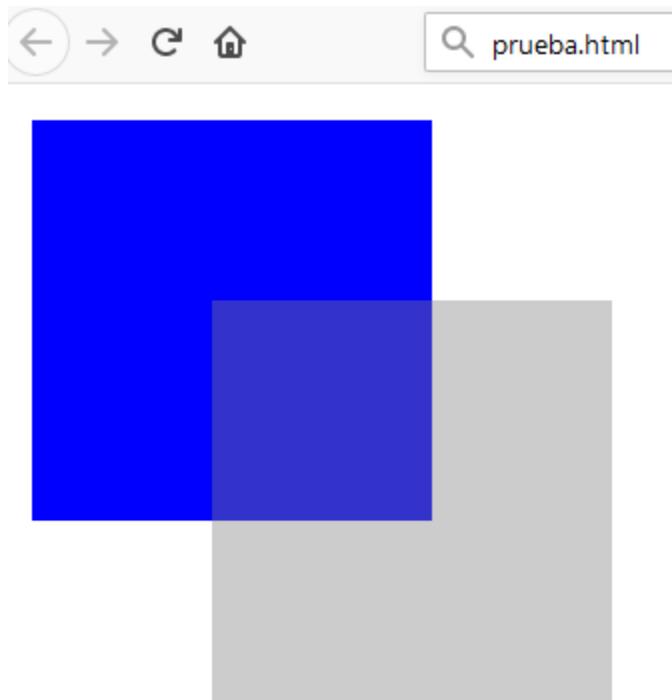


Ilustración 300: Rectángulo semitransparente

## Referencias

- [1] W3Schools, «JavaScript Tutorial,» enero 2019. [En línea]. Available: <https://www.w3schools.com/js/>. [Último acceso: 17 enero 2019].
- [2] Mozilla, «JavaScript,» enero 2019. [En línea]. Available: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. [Último acceso: 17 enero 2019].
- [3] W3Techs, «Usage of client-side programming languages for websites,» enero 2019. [En línea]. Available: [https://w3techs.com/technologies/overview/client\\_side\\_language/all](https://w3techs.com/technologies/overview/client_side_language/all). [Último acceso: 08 enero 2019].
- [4] Adobe, «Flash,» enero 2019. [En línea]. Available: <https://get.adobe.com/es/flashplayer/>. [Último acceso: 17 enero 2019].
- [5] UnoCero, «Adobe anuncia el esperado final de Flash,» julio 2017. [En línea]. Available: <https://www.unocero.com/entretenimiento/adobe-anuncia-esperado-final-flash/>. [Último acceso: 29 noviembre 2018].
- [6] W3Schools, «ECMAScript 6 - ECMAScript 2015,» enero 2019. [En línea]. Available: [https://www.w3schools.com/js/js\\_es6.asp](https://www.w3schools.com/js/js_es6.asp). [Último acceso: 17 enero 2019].
- [7] ecma international, «ECMAScript® 2015 Language Specification,» 2015. [En línea]. Available: <http://www.ecma-international.org/ecma-262/6.0/>. [Último acceso: 17 enero 2019].
- [8] W3Schools, «JavaScript versions,» [En línea]. Available: [https://www.w3schools.com/js/js\\_versions.asp](https://www.w3schools.com/js/js_versions.asp). [Último acceso: 17 enero 2019].