

Autor: Rafael Alberto Moreno Parra

Capacitándose en JAVASCRIPT

2025

Contenido

Tabla de Ilustraciones	7
Otros libros del autor.....	9
Página web del autor y canal en Youtube	11
Sitio en GitHub	11
Licencia de este libro	12
Licencia del software	12
Marcas registradas	12
Introducción.....	13
Iniciando	14
Navegadores	14
Editores de texto.....	14
Código fuente	14
Preparando el entorno para trabajar con JavaScript	15
Probando que el servidor de aplicaciones funciona bien	21
Iniciando con JavaScript	22
Comentarios en el código.....	23
El tradicional “Hola Mundo”	23
¿Dónde guardar los archivos HTML?	24
Uso de variables	28
Operaciones matemáticas	30
Asignación	31
Operaciones con cadenas.....	32
Impresión de caracteres propios del idioma español	35
Constantes matemáticas.....	37
Funciones trigonométricas	38
Funciones Matemáticas.....	39
Captura de datos tipo texto por pantalla	40
Captura de datos tipo numérico por pantalla	41
Ejemplos de cálculos	42
Corrigiendo errores en JavaScript	43
Validación en línea del código	45
Sí condicional	48
Operadores lógicos: Conjunción (Y, &&) y Disyunción (O,).....	50
Uso del switch	51
Ciclos. Uso del “for”	52
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	71
Método de Bisección	76
Ámbito (scope) de las variables y funciones	77
Funciones recursivas	81
Uso de eval como evaluador de expresiones algebraicas	83
Números aleatorios	84
Distribución Normal.....	85
Distribución Triangular.....	86

Distribución Uniforme	87
Manejo de errores con números.....	88
isNaN	88
isFinite.....	89
Try...catch	90
Arreglos unidimensionales o vectores	91
Borrar elementos con splice()	93
Operaciones con arreglos	94
Ordenación de arreglo unidimensional	96
Funciones genéricas para arreglos.....	99
Retornar arreglos desde funciones.....	100
Implementación de algoritmos de ordenación	101
Algoritmo de burbuja.....	101
Algoritmo de ordenación por selección	103
Algoritmo de ordenación por inserción	104
Algoritmo de ordenación QuickSort.....	105
Algoritmo de ordenación MergeSort.....	106
Manejo de Fechas	107
Manejo de cadenas o strings	108
Ejemplos de programas usando cadenas.....	110
Invertir Cadena.....	110
Quitar los espacios de una cadena.....	111
Quitar las vocales de una cadena	112
Quitar caracteres no permitidos	113
Un sencillo cifrado / descifrado	114
Un sencillo cifrado / descifrado con clave de cifrado.....	115
Sumar largos números enteros almacenados en cadenas	116
Arreglos bidimensionales	117
Con números.....	117
Con caracteres.....	117
Función genérica	118
Funciones genéricas	118
Ejemplos de programas usando arreglos bidimensionales.....	119
Poner una torre al azar en un tablero de ajedrez y mostrar su ataque.....	119
Poner un Rey al azar en un tablero de ajedrez y mostrar su ataque.....	120
Poner un alfil al azar en un tablero de ajedrez y mostrar su ataque	121
Poner una Reina al azar en un tablero de ajedrez y mostrar su ataque.....	122
Ruta del menor costo.....	123
Resolver Sudokus.....	125
Poner los barcos en el juego batalla naval	127
Funciones que reciben distinto número de parámetros.....	129
Funciones que reciben a su vez funciones por parámetros.....	130
Operaciones de bit	132
Convertir un entero a su representación binaria	132
Operaciones OR, AND y XOR directas	133
Multiplicar usando operaciones de bit.....	136
Dividir usando operaciones de bit.....	137
Intercambiar valores usando operaciones de bit.....	138
Programación orientada a objetos	139
Definiendo clases, constructores y atributos	139

Definiendo métodos.....	140
Métodos y atributos.....	141
Atributos y métodos privados	142
Getters y Setters	143
Herencia	144
Herencia y constructor.....	145
Sobrecarga de métodos	147
Uso de typeof para detectar el tipo de dato de una variable	148
Árbol binario	149
Recorrido en pre-orden	150
Recorrido en in-orden	151
Recorrido en post-orden.....	152
Ordenación usando un árbol binario.....	153
Lista de objetos. Pilas.....	154
Lista de objetos	155
Instrucciones que ejecutan controlando el tiempo	156
Uso de JSON	158
Control sobre la página HTML	162
Captura el evento de dar clic en un <div>.....	164
Captura el evento de dar doble clic en un <div>	166
Captura el evento de pasar el cursor del ratón por encima de un <div>	167
Captura el evento de mover el cursor del ratón fuera del <div>.....	168
Captura el evento apenas cargue la página	169
Captura el evento cuando de clic al interior del <div>	170
Captura el evento al soltar el botón del ratón dentro del <div>.....	171
Captura el evento de mover el cursor del ratón.....	172
Captura el evento cuando se cambia el tamaño de la ventana.....	173
Captura el evento de dar “submit” en un formulario	174
Captura el evento de entrar a la caja de texto	175
Captura el evento al salir de una caja de texto.....	176
Captura el evento al presionar una tecla dentro de una caja de texto	177
Captura el evento cuando suelta la tecla	179
Captura el evento de cambio del texto en una caja de texto	180
Mostrar el código de la tecla presionada.....	181
Pregunta antes de ir a la página enlazada	182
DOM (Document Object Model).....	183
Cambiar atributos de un objeto.....	183
Cambiar el color de fondo de un <div>	183
Hacer visible o no un <div>	184
Poner o quitar texto de un <div>	185
Cambiar el aspecto del objeto cuando está seleccionado	186
Validar valores al saltar a otro objeto	187
Contar objetos	188
Obtener lista de objetos	189
Obtener determinado objeto de una lista.....	190
Obtener lista de determinado tipo de objetos y ver el valor de sus atributos.....	191
Obtener la lista de determinado tipo de objetos y cambiar sus atributos	192
Mostar la ubicación del documento	193
Cambiar el estilo CSS de un determinado objeto	194
Mostrar la codificación del documento.....	195

Mostrar el título del documento.....	196
Saltos de línea sin usar 	197
Mostrar la fecha de modificación del documento.....	198
Mostrar el tipo de documento	199
Mostrar el dominio del documento.....	200
Mostrar el estado del documento.....	201
Mostrar las dimensiones del documento	202
Mostrar el número de enlaces	203
Mostrar hacia donde apuntan los enlaces	204
Crear botones en tiempo de ejecución	205
Crear <p> en tiempo de ejecución	206
Crear nuevos <p> y contarlos en tiempo de ejecución.....	207
Contar el número de objetos en el documento	208
Contar el número de formularios en el documento	209
Contar el número de imágenes del documento.....	210
Describir los objetos de la página	211
Abrir una ventana	212
Abrir un conjunto de ventanas	213
Un formulario que ejecuta un algoritmo local al enviar la información	214
Valida un correo con una expresión regular	215
Valida una URL con expresiones regulares.....	216
Gráficos	217
Dibujar líneas	219
Cambiar el grueso de una línea	220
Dar un color determinado a una línea	221
Color RGB	222
Estilos para finalizar las líneas	223
Dibujar varias líneas.....	224
Juntar líneas.....	225
Probando los otros estilos de juntar líneas.....	226
Varias líneas.....	227
Genera una figura cerrada y la rellena	228
Dibujar varias líneas horizontales con un ciclo	229
Dibujar varias líneas verticales con un ciclo	230
Ilusión de curva con varias líneas.....	231
Dibujar círculos.....	233
Dibujar curvas	235
Curva de Bézier	236
Combinando curvas.....	237
Rectángulo y gradiente de color	238
Variando en forma oblicua con varios colores.....	239
Variando en forma vertical	240
Variando en forma horizontal.....	241
Variando en círculos	242
Trabajando con imágenes	243
Variar posición y tamaño de la imagen.....	244
Convertir imagen a escala de grises	245
Dibujar letras	247
Sombras	252
Semitransparente	253

Librería Gráfica	254
Referencias	274

Tabla de Ilustraciones

Ilustración 1: Lenguajes de programación para aplicaciones Web en el lado del cliente	13
Ilustración 2: Sitio web para la descarga de XAMPP	15
Ilustración 3: Pantalla de advertencia sobre el User Account Control	15
Ilustración 4: Pantalla inicial del instalador.....	16
Ilustración 5: Solo Apache, MySQL, PHP y phpMyAdmin.....	16
Ilustración 6: El directorio donde se instalará XAMPP.....	17
Ilustración 7: Idioma inglés.....	17
Ilustración 8: Listo para instalar	17
Ilustración 9: Instalando XAMPP	18
Ilustración 10: Apache HTTP Server requiere permisos del Firewall, se presiona "Permitir acceso"	18
Ilustración 11: Finaliza la instalación	18
Ilustración 12: Panel de control de XAMPP, los servicios están inactivos.....	19
Ilustración 13: Iniciando el servidor web Apache.	19
Ilustración 14: MySQL requiere permisos en el firewall, clic en "Permitir acceso".	19
Ilustración 15: Los dos servicios están activos.....	20
Ilustración 16: ¿Apache funciona? abre un navegador y se escribe http://localhost.....	21
Ilustración 17: Notepad++ en español	22
Ilustración 18: Configuramos Notepad++ para editar archivos HTML y JavaScript.....	22
Ilustración 19: Botón de guardar	23
Ilustración 20: En la carpeta XAMPP dentro de HTDOCS se crea SRC.....	24
Ilustración 21: Guardar la página HTML en el PC.....	24
Ilustración 22: Hay que fijarse que el archivo tenga extensión .html	25
Ilustración 23: El directorio de los ejemplos de JavaScript	26
Ilustración 24: Ejecución del programa.....	26
Ilustración 25: Impresión del valor de cada variable.....	28
Ilustración 26: Primero se hace la operación matemática y luego la concatenación	33
Ilustración 27: Se hace la operación matemática y el resto es concatenación	34
Ilustración 28: Imprimir caracteres propios del idioma español.....	35
Ilustración 29: En Chrome puede pasar que no se impriman	35
Ilustración 30: Funciona en Mozilla Firefox	35
Ilustración 31: Ante un fallo así, verificamos la codificación. Usualmente está en "Codificar en ANSI"	35
Ilustración 32: Seleccionamos la opción de "Convertir a UTF-8"	36
Ilustración 33: Después de guardar, se chequea que esté en la opción "Codificar en UTF-8".....	36
Ilustración 34: Cuadro de diálogo para capturar un valor texto	40
Ilustración 35: Muestra el valor capturado	40
Ilustración 36: Código en JavaScript con varios errores	43
Ilustración 37: Ante errores en JavaScript, el navegador queda en blanco.....	43
Ilustración 38: Presionando F12 en el navegador Edge y dando clic en Consola se muestra el error.....	44
Ilustración 39: JavaScript Validator para chequear sintaxis	45
Ilustración 40: Diagnóstico de JavaScript Validator de un script de ejemplo	46
Ilustración 41:JavaScript Tester para validar sintaxis	46
Ilustración 42: Jshint para validar sintaxis	47
Ilustración 43: Ámbito (scope) de una variable.....	77
Ilustración 44: Ámbito (scope) de una variable.....	78
Ilustración 45: Ámbito (scope) de una variable.....	79
Ilustración 46: Ámbito (scope) de una variable.....	80
Ilustración 47: Número aleatorio	84
Ilustración 48: Ingresa expresión con sintaxis errónea	90
Ilustración 49: Mensaje de error por la sintaxis errónea.....	90
Ilustración 50: Ruta del menor costo. Genera nuevas rutas más económicas.....	124
Ilustración 51: Sudoku resuelto	126
Ilustración 52: Barcos ubicados en el juego de mesa Batalla Naval	128
Ilustración 53: Valores de atributos.....	139
Ilustración 54: Métodos privados	142
Ilustración 55: Árbol binario	150
Ilustración 56: Botón.....	162
Ilustración 57: Ventana emergente generada por JavaScript	162
Ilustración 58: Ventana emergente al dar clic en un hipervínculo	163
Ilustración 59: Ventana emergente al dar clic en un <div>	164
Ilustración 60: Ventana emergente al dar clic en un <div>	165
Ilustración 61:Lienzo ("canvas") visible usando un borde	218
Ilustración 62: Línea dibujada	219
Ilustración 63: Línea gruesa	220

Ilustración 64: Color a la línea.....	221
Ilustración 65: Línea con color modificando los parámetros RGB	222
Ilustración 66: Las líneas finalizan redondeadas en ambos extremos usando "round"	223
Ilustración 67:Tres líneas con la misma longitud, pero diferente cierre de extremo	224
Ilustración 68: Dos líneas juntas con un empalme redondeado	225
Ilustración 69: Diferentes formas de juntar dos líneas: "round", "miter", "bevel"	226
Ilustración 70: Una figura cerrada.....	227
Ilustración 71: Figura cerrada y rellena de un color.....	228
Ilustración 72: Líneas generadas por un ciclo	229
Ilustración 73: Líneas verticales dibujadas con un ciclo.....	230
Ilustración 74: Ilusión de curva usando líneas rectas	231
Ilustración 75: Ilusión de dos curvas usando líneas rectas	232
Ilustración 76: La manera en que se dibujará el círculo	233
Ilustración 77: Círculo	234
Ilustración 78: Forma en que se dibujarán las curvas	235
Ilustración 79: Curva.....	235
Ilustración 80: Curva Bézier.....	236
Ilustración 81: Combinando curvas	237
Ilustración 82: Variación del color	238
Ilustración 83: Variación de varios colores en forma oblicua	239
Ilustración 84: Variación de color en forma vertical	240
Ilustración 85: Variación de color horizontal	241
Ilustración 86: Variación en círculos	242
Ilustración 87: La imagen rellena el rectángulo repitiéndose.....	243
Ilustración 88: Imagen original.....	245
Ilustración 89: Imagen en escala de grises	246
Ilustración 90: Dibuja un texto	247
Ilustración 91: Dibuja un texto con relleno de color.....	248
Ilustración 92: Texto dibujado, se muestra el borde de cada letra.....	249
Ilustración 93: Dibuja un texto alineado al final	250
Ilustración 94: Muestra el ancho del texto dibujado.....	251
Ilustración 95: Sombra	252
Ilustración 96: Rectángulo semitransparente.....	253
Ilustración 97: Uso de librería gráfica	267
Ilustración 98: Uso de librería gráfica	268
Ilustración 99: Uso de librería gráfica	269
Ilustración 100: Uso de librería gráfica	270
Ilustración 101: Uso de librería gráfica	271
Ilustración 102: Uso de librería gráfica	272
Ilustración 103: Uso de librería gráfica	273

Otros libros del autor

Libro 21: Dividido en varias partes, cada uno es un libro

C# y .NET 9. 2025.

- Parte 1: IDE, variables, ciclos, si condicional, funciones
- Parte 2: Cadenas (strings)
- Parte 3: Arreglos estáticos
- Parte 4: Programación Orientada a Objetos
- Parte 5: Estructuras de datos dinámicas
- Parte 6: LINQ y Lambda
- Parte 7: Un evaluador de expresiones algebraicas
- Parte 8: Estructuras de datos de bajo nivel
- Parte 9: Simulaciones
- Parte 10: Algoritmos evolutivos
- Parte 11: Redes Neuronales
- Parte 12: Gráficos 2D
- Parte 13: Gráficos en 3D
- Parte 14: Animación
- Parte 15: Imágenes

En Colombia 2025. Págs. 1446. Libro y código fuente descargable en: <https://github.com/ramsoftware/C-Sharp>

Libro 20: "Gráficos en C#". En Colombia 2022. Págs. 213. Libro y código fuente descargable en:

<https://github.com/ramsoftware/C-Sharp-Graficos>. **Obsoleto, se retira del repositorio.**

Libro 19: "Evaluador de expresiones en nueve lenguajes de programación. En C#, C++, Delphi, Java, JavaScript, PHP, Python, TypeScript y Visual Basic .NET". En Colombia 2021. Págs. 158. Libro y código fuente descargable en:

<https://github.com/ramsoftware/Evaluador3>

Libro 18: "C#: Árboles binarios, n-arios, grafos, listas simple y doblemente enlazadas". En Colombia 2021. Págs. 63. Libro y código fuente descargable en: <https://github.com/ramsoftware/C-Sharp-Arboles>. **Obsoleto, se retira del repositorio.**

Libro 17: "C#. Estructuras dinámicas de memoria". En Colombia 2021. Págs. 82. Libro y código fuente descargable en: <https://github.com/ramsoftware/CSharpDinamica>. **Obsoleto, se retira del repositorio.**

Libro 16: "C#. Programación Orientada a Objetos". En Colombia 2020. Págs. 90. Libro y código fuente descargable en: <https://github.com/ramsoftware/C-Sharp-POO>. **Obsoleto, se retira del repositorio.**

Libro 15: "C#. Estructuras básicas de memoria". En Colombia 2020. Págs. 60. Libro y código fuente descargable en: <https://github.com/ramsoftware/EstructuraBasicaMemoriaCSharp>. **Obsoleto, se retira del repositorio.**

Libro 14: "Iniciando en C#". En Colombia 2020. Págs. 72. Libro y código fuente descargable en: <https://github.com/ramsoftware/C-Sharp-Iniciando>. **Obsoleto, se retira del repositorio.**

Libro 13: "Algoritmos Genéticos". En Colombia 2020. Págs. 62. Libro y código fuente descargable en: <https://github.com/ramsoftware/LibroAlgoritmoGenetico2020>. **Obsoleto, se retira del repositorio.**

Libro 12: "Redes Neuronales. Segunda Edición". En Colombia 2020. Págs. 108. Libro y código fuente descargable en: <https://github.com/ramsoftware/LibroRedNeuronal2020>. **Obsoleto, se retira del repositorio.**

Libro 11: "Capacitándose en JavaScript". En Colombia 2020. Págs. 317. **Obsoleto, se retira del repositorio.**

Libro 10: "Desarrollo de aplicaciones para Android usando MIT App Inventor 2". En Colombia 2016. Págs. 102. **Obsoleto, se retira del repositorio.**

Libro 9: "Redes Neuronales. Parte 1.". En Colombia 2016. Págs. 90. **Obsoleto, se retira del repositorio.**

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. **Obsoleto, se retira del repositorio.**

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. **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> . **Obsoleto (No hay enlace).**

Libro 3: "Simulación: Conceptos y Programación" En Colombia 2012. Págs. 81. Ubicado en: <https://openlibra.com/es/book/simulacion-conceptos-y-programacion> . **Obsoleto (No hay enlace).**

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 principalmente 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 2025 de cómo está repartido los lenguajes de programación en el cliente [3]:

El enlace es: https://w3techs.com/technologies/overview/client_side_language/all

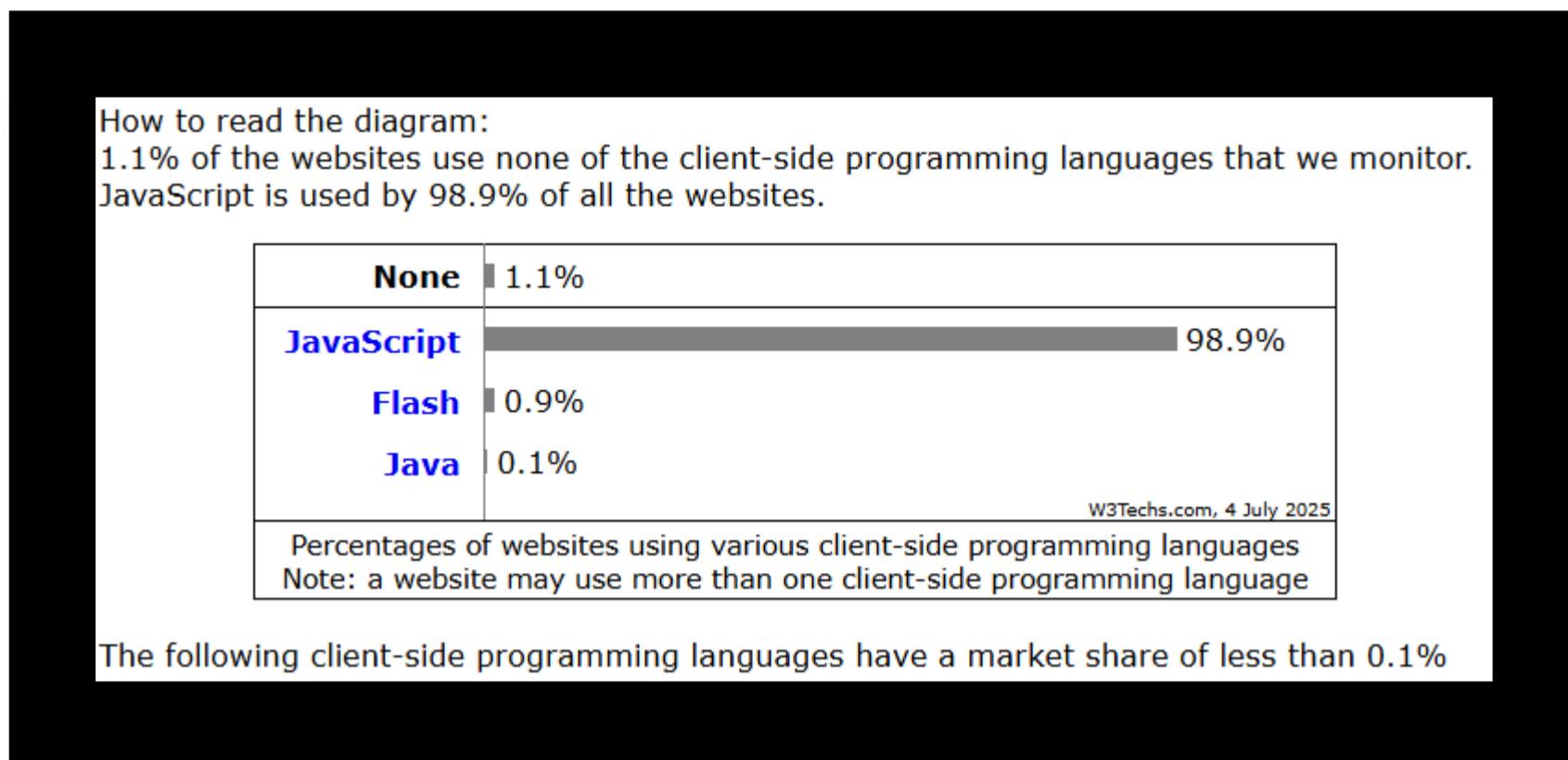


Ilustración 1: Lenguajes de programación para aplicaciones Web en el lado del cliente

JavaScript es el lenguaje de facto, además, se añade que Flash [4] es una tecnología que está abandonada por la empresa que la creó.

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 2025 [5]

Iniciando

Empezar a programar en JavaScript es fácil y gratuito. Sólo es requerido tener un navegador moderno instalado en el PC (Microsoft Edge, Mozilla Firefox, Opera, Google Chrome, Brave, Vivaldi) y un editor de texto (recomendado Notepad++ que es gratuito). Este manual mostrará capturas de pantalla en el ambiente de Microsoft Windows 11, 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 Vivaldi: <https://vivaldi.com/es/>

Enlace para Brave: <https://brave.com/es/>

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

https://www.w3schools.com/js/js_2025.asp [6]

Editores de texto

Enlace para Notepad++: <https://notepad-plus-plus.org/downloads/>

Código fuente

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

Preparando el entorno para trabajar con JavaScript

Es posible trabajar con JavaScript directamente con el navegador: Se escribe el código de JavaScript en un archivo con extensión HTML y se ejecuta en el navegador, pero esta técnica conlleva el riesgo de que el desarrollador se equivoque con el uso de las rutas y que al final se le complique poder llevar el programa a una aplicación Web que es el objetivo final: Hacer una aplicación Web dinámica.

Luego la forma más recomendada para trabajar con JavaScript es instalando un servidor de aplicaciones, el cual tiene los siguientes componentes:

1. Un servidor Web [7]
2. Un gestor de base de datos [8]

En el caso del servidor Web se hace uso de Apache y el gestor de base de datos es MariaDB [9] o MySQL [10] (ambos son compatibles). Los dos son software libre, no hay que pagar un licenciamiento por usarlos.

Los dos: Apache y MariaDB/MySQL se pueden instalar por aparte, directamente de los sitios oficiales, pero hay una manera mucho más rápida y fácil y es usando una aplicación que instala esos componentes y los configura automáticamente. Uno muy conocido es XAMPP.

Instalar XAMPP es sencillo, se mostrará paso a paso desde su descarga hasta su instalación y ejecución. El primer paso es ir a <https://www.apachefriends.org/download.html> y descargar el instalador.



Ilustración 2: Sitio web para la descarga de XAMPP

Inicia la instalación

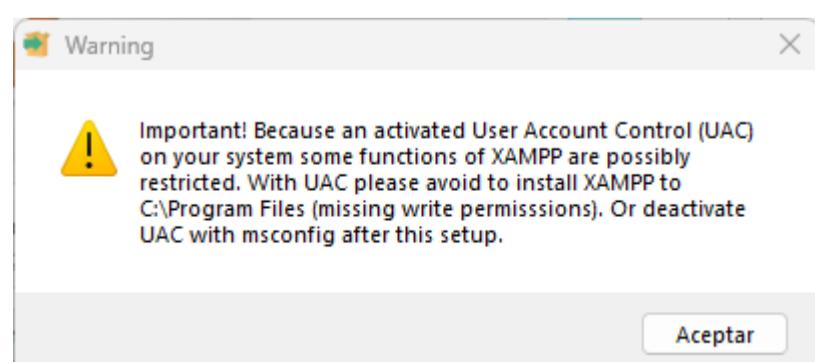


Ilustración 3: Pantalla de advertencia sobre el User Account Control

UAC significa Control de Cuentas de Usuario y es para que las aplicaciones ejecuten en un modo NO administrador del sistema. XAMPP requiere tener acceso de administrador, luego Windows 11 preguntará si va a ser instalarlo, se responde afirmativamente.

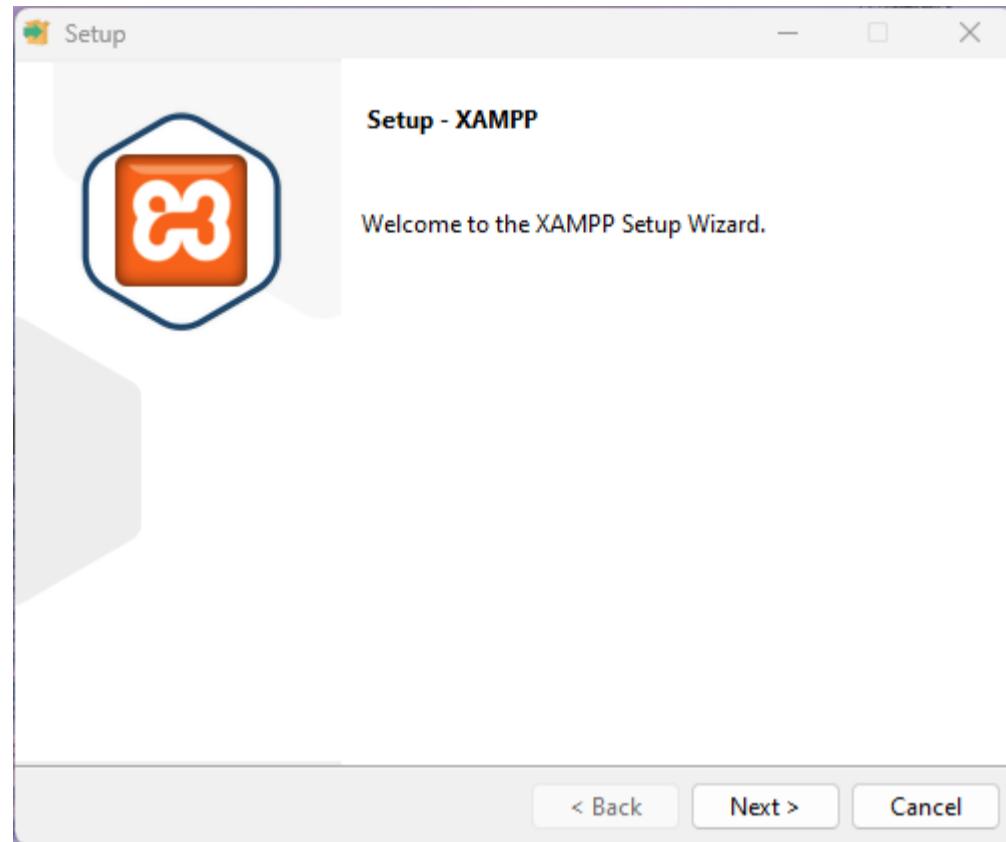


Ilustración 4: Pantalla inicial del instalador.

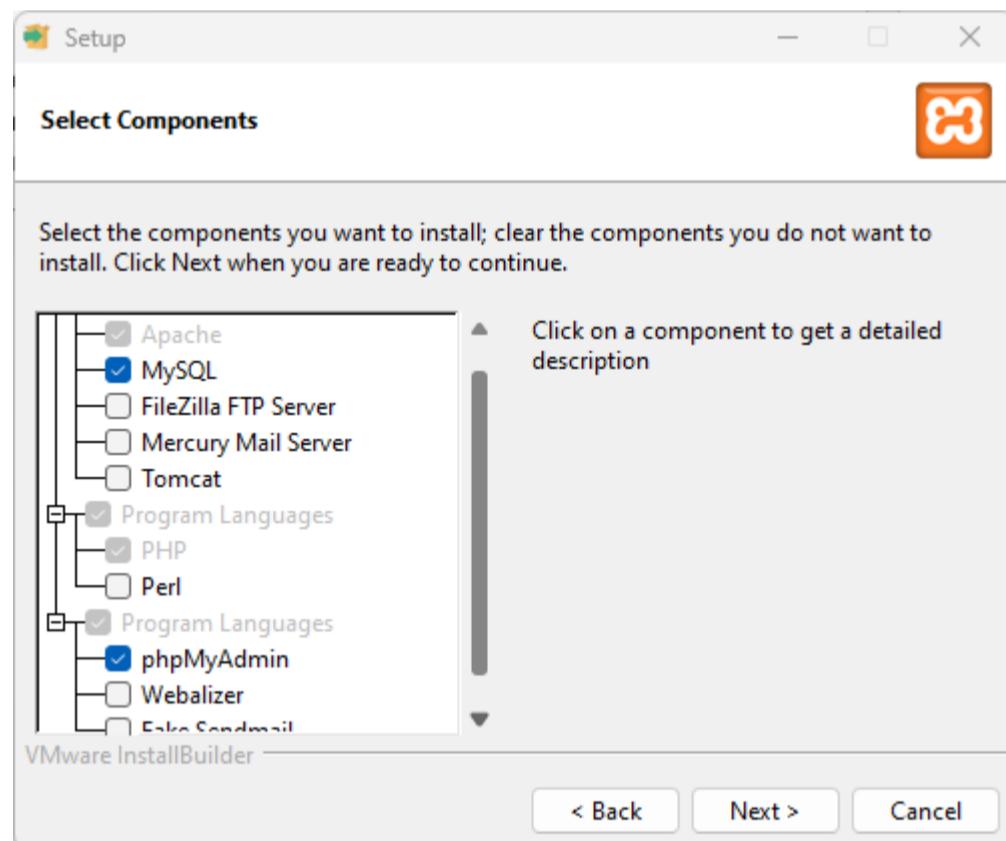


Ilustración 5: Solo Apache, MySQL, PHP y phpMyAdmin.

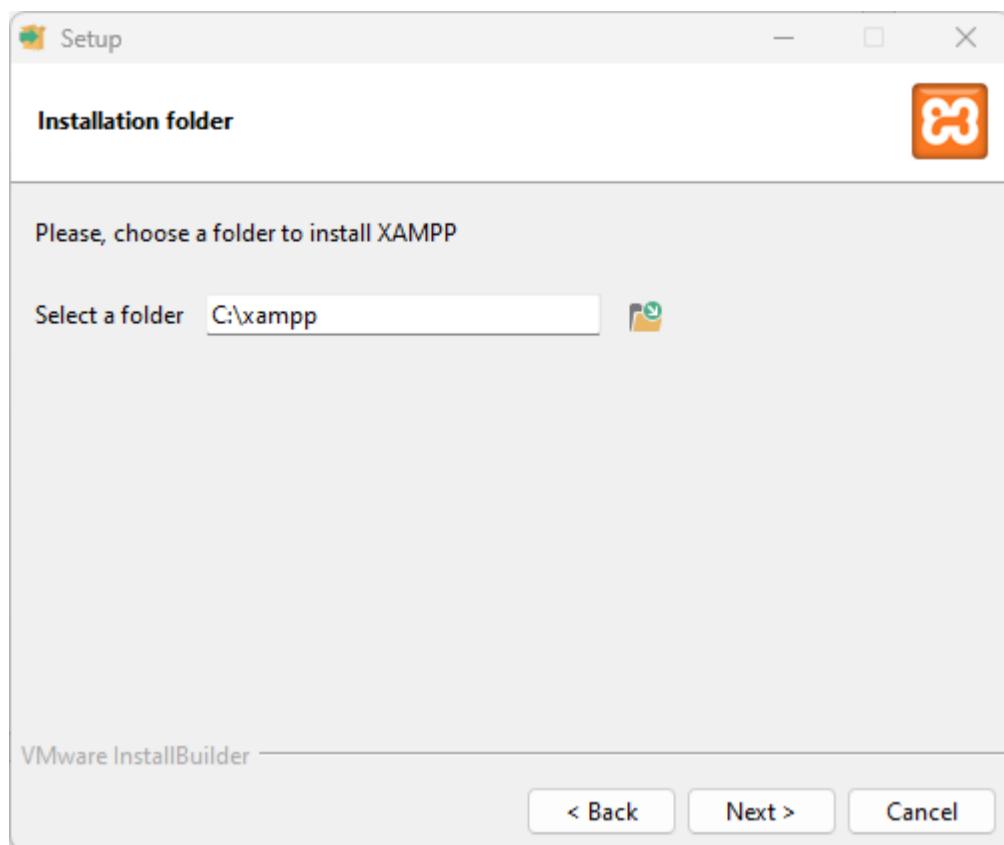


Ilustración 6: El directorio donde se instalará XAMPP.

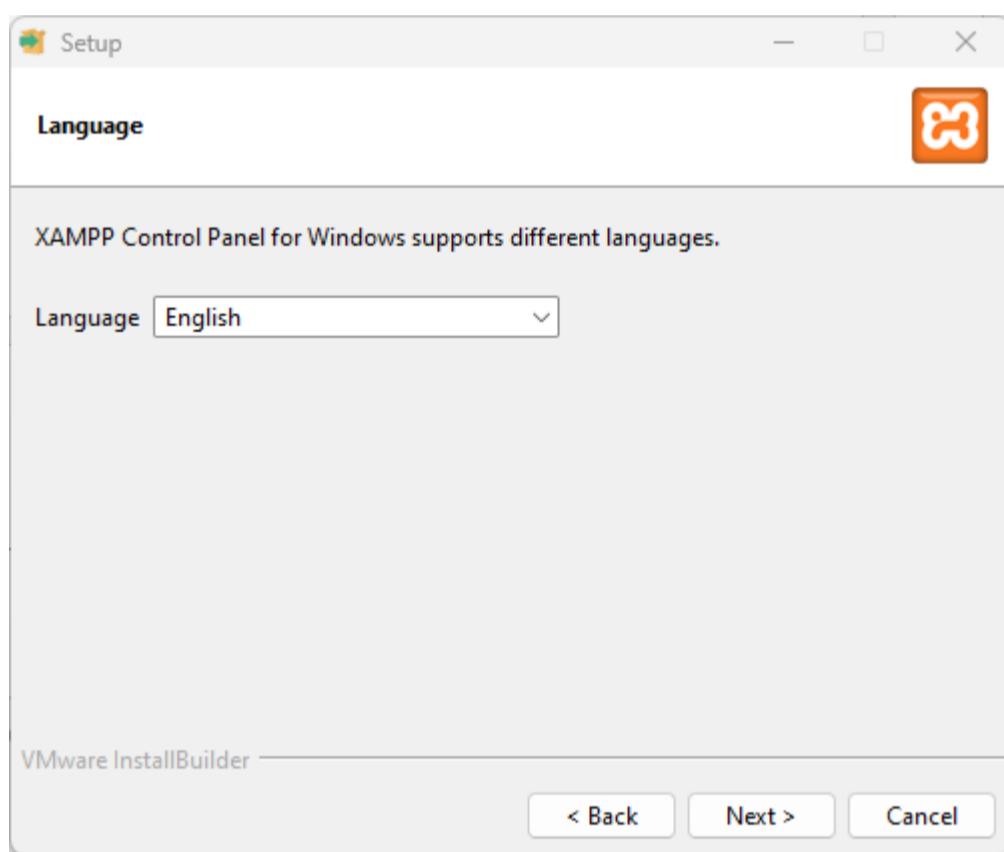


Ilustración 7: Idioma inglés

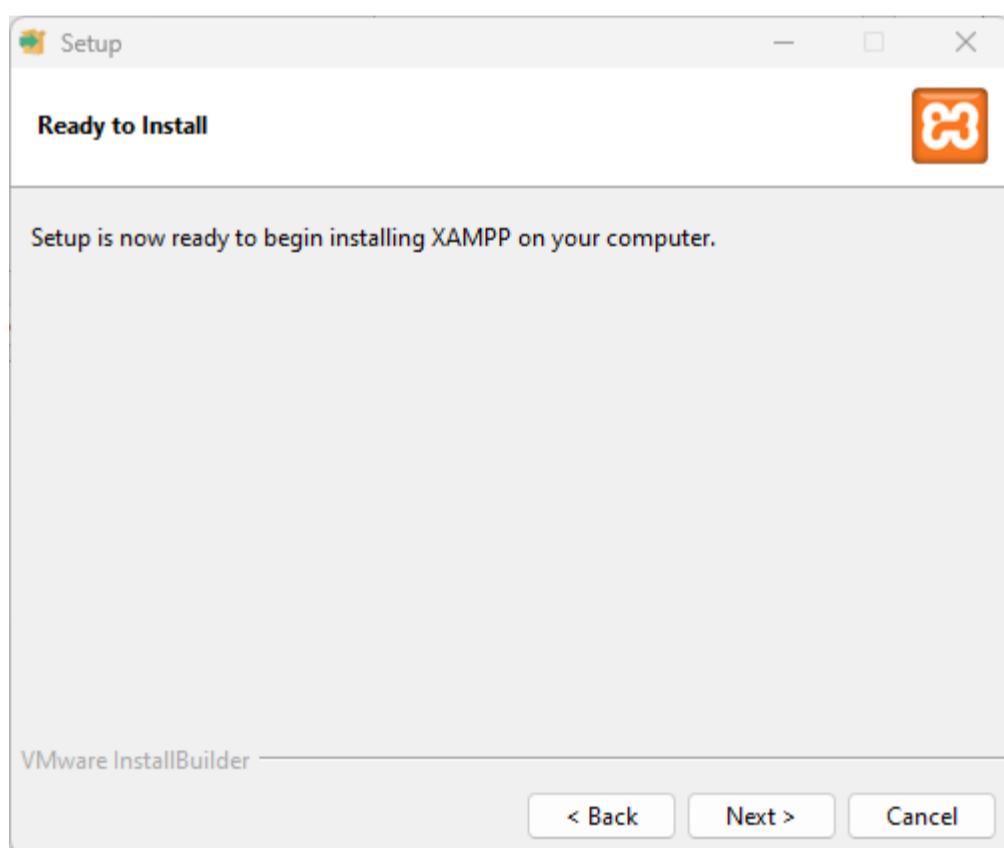


Ilustración 8: Listo para instalar

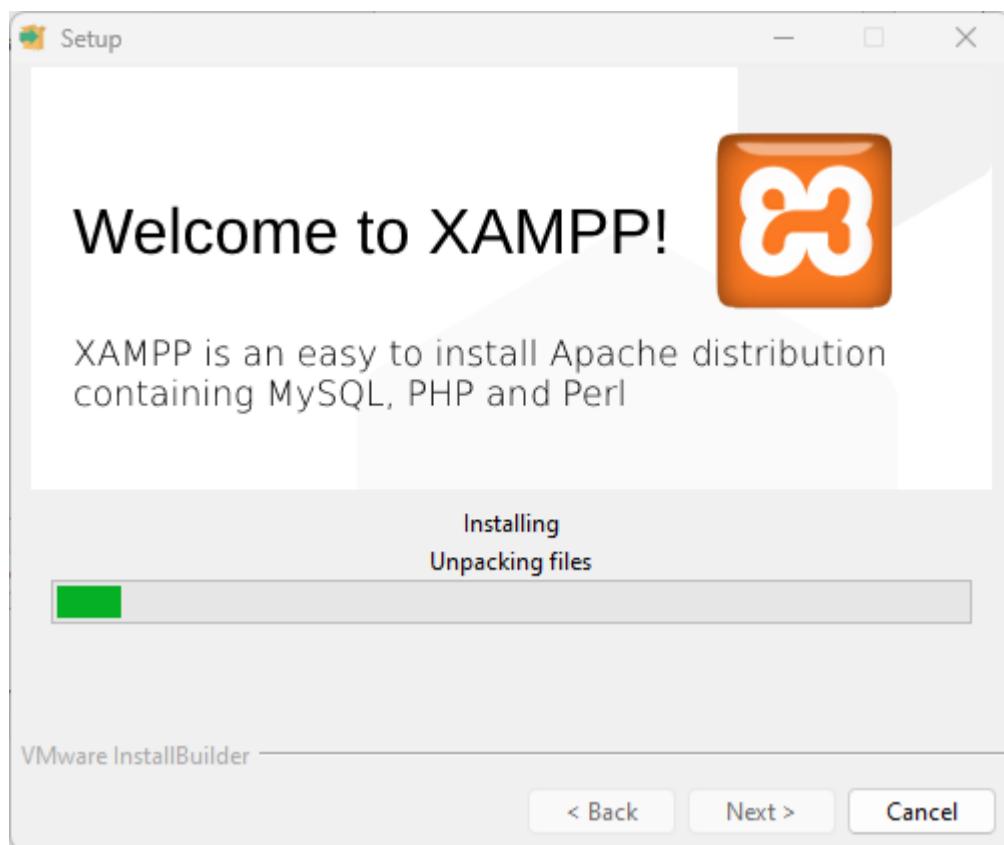


Ilustración 9: Instalando XAMPP

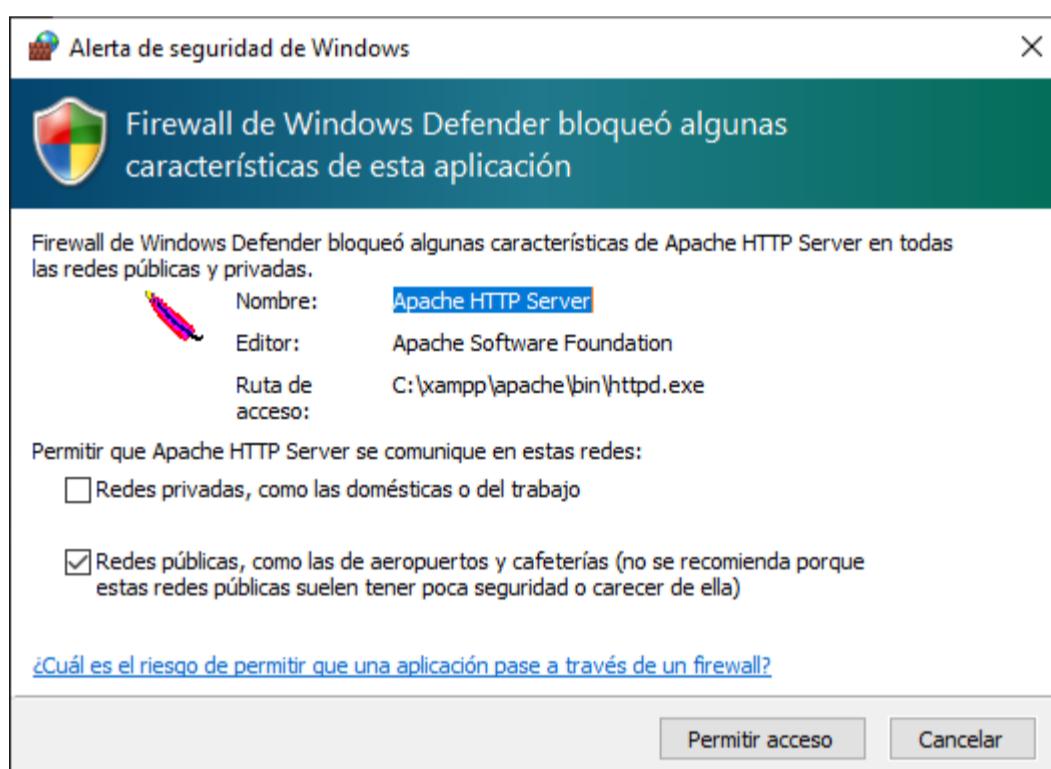


Ilustración 10: Apache HTTP Server requiere permisos del Firewall, se presiona "Permitir acceso"

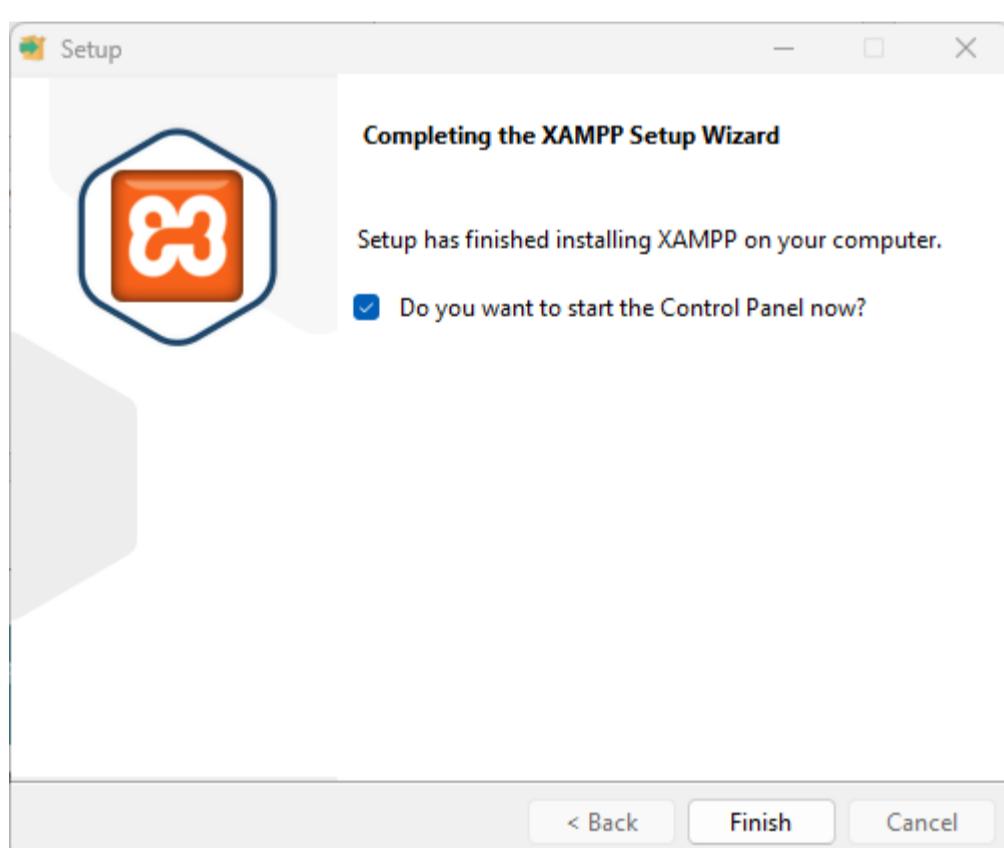


Ilustración 11: Finaliza la instalación

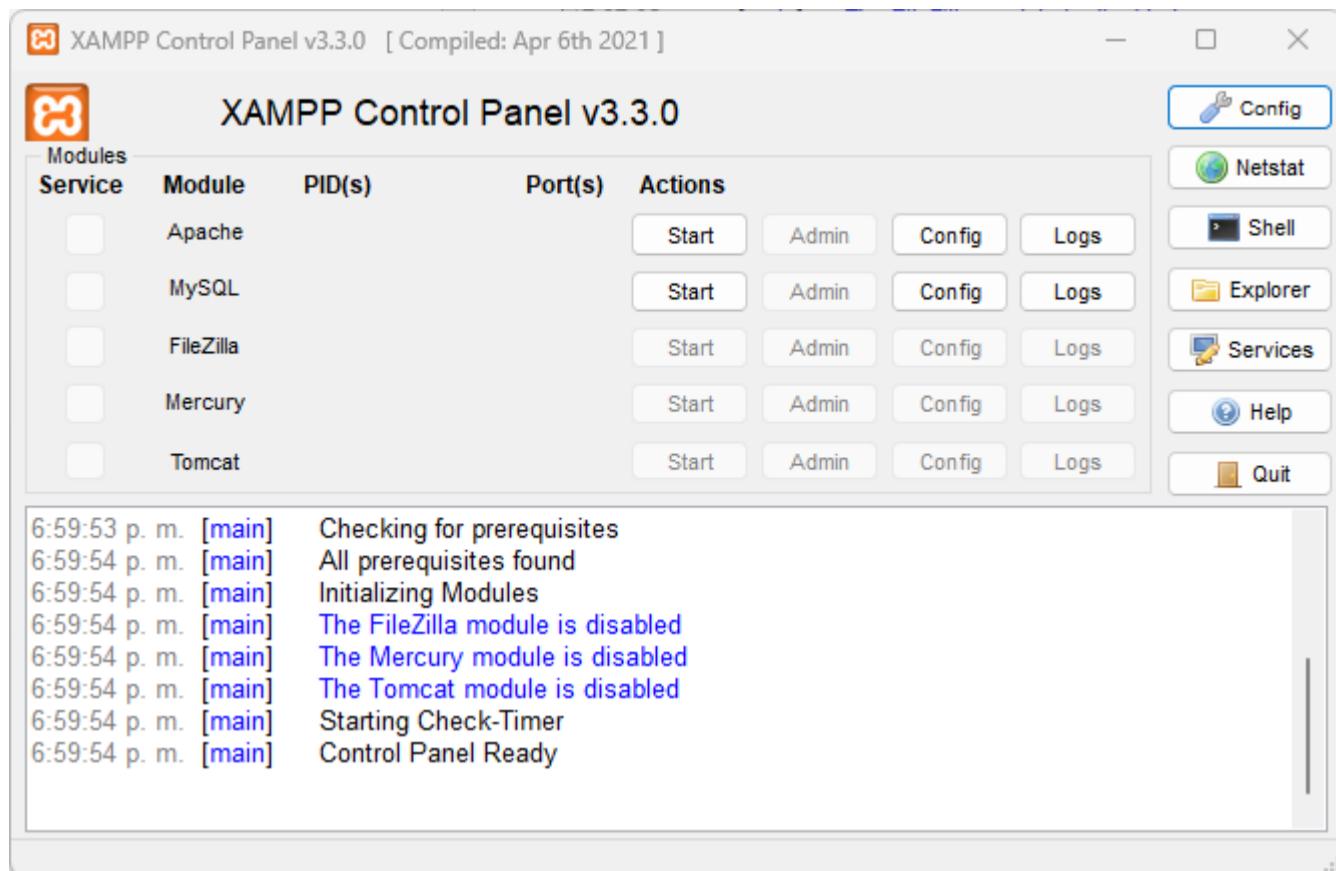


Ilustración 12: Panel de control de XAMPP, los servicios están inactivos.

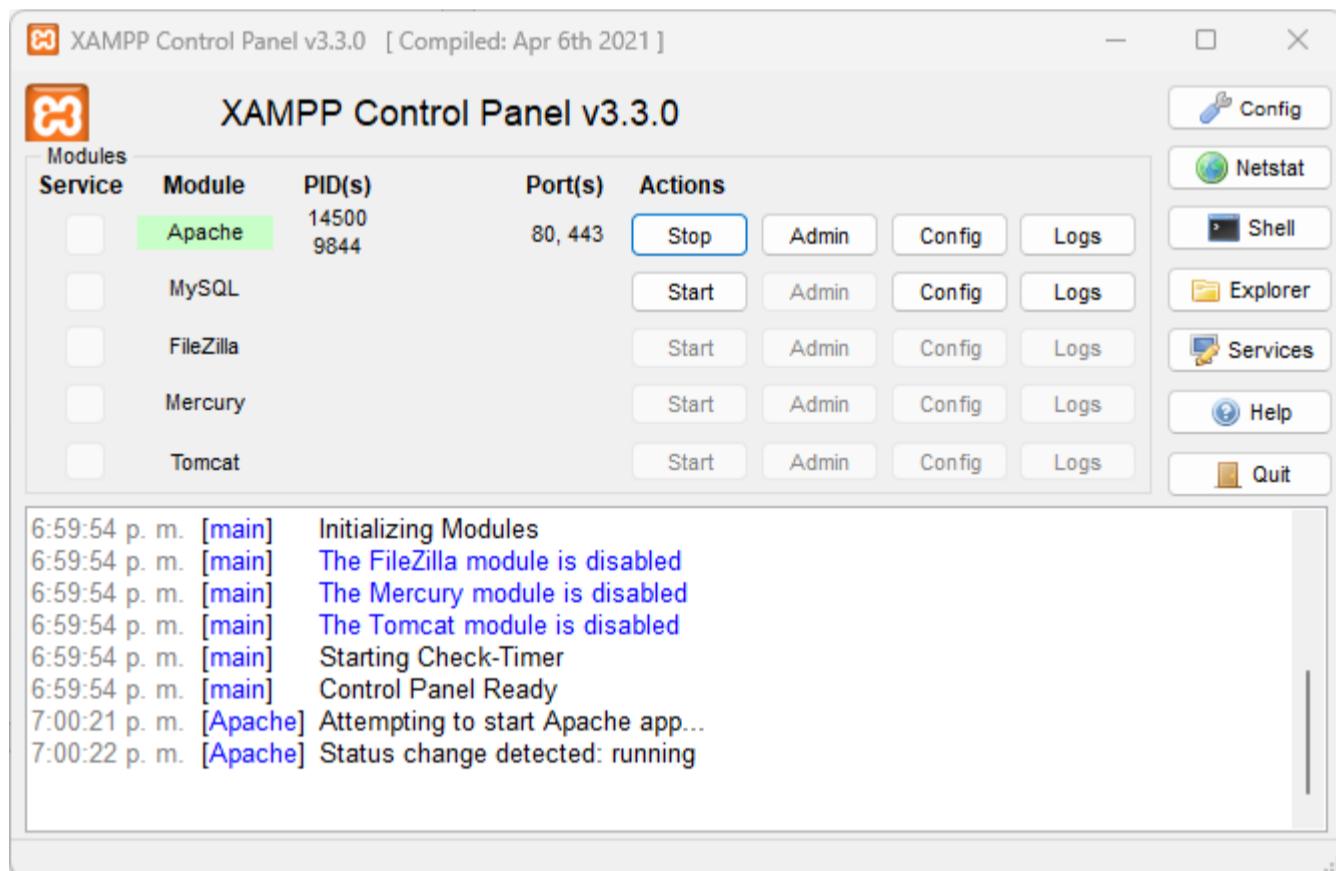


Ilustración 13: Iniciando el servidor web Apache.

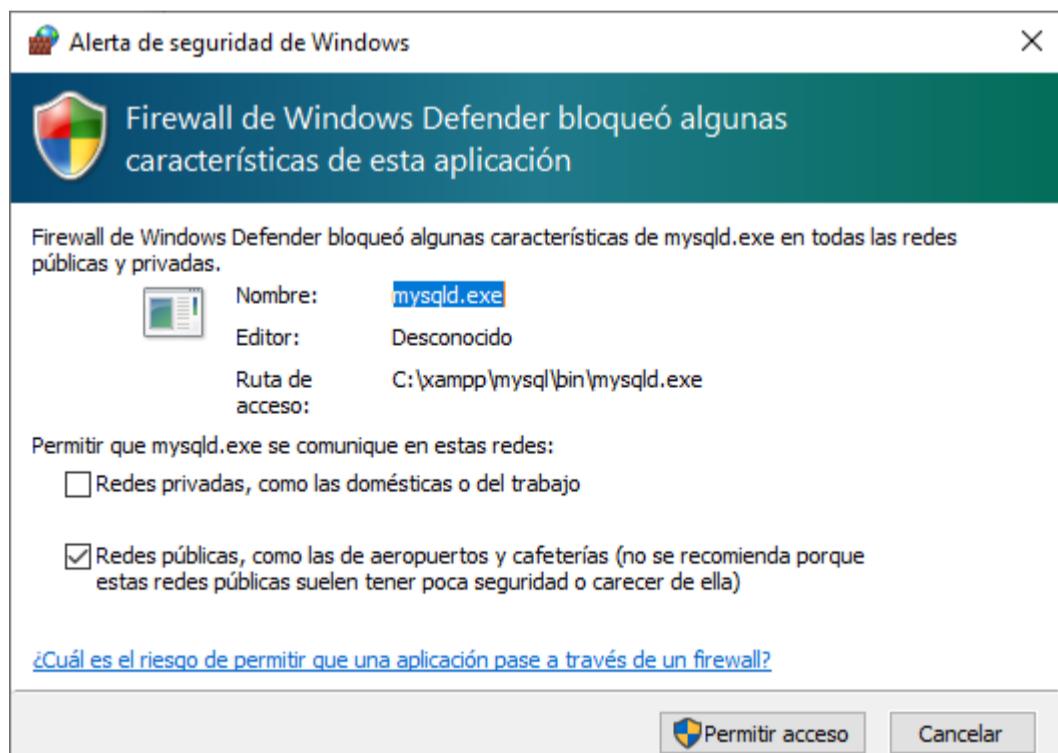


Ilustración 14: MySQL requiere permisos en el firewall, clic en "Permitir acceso".

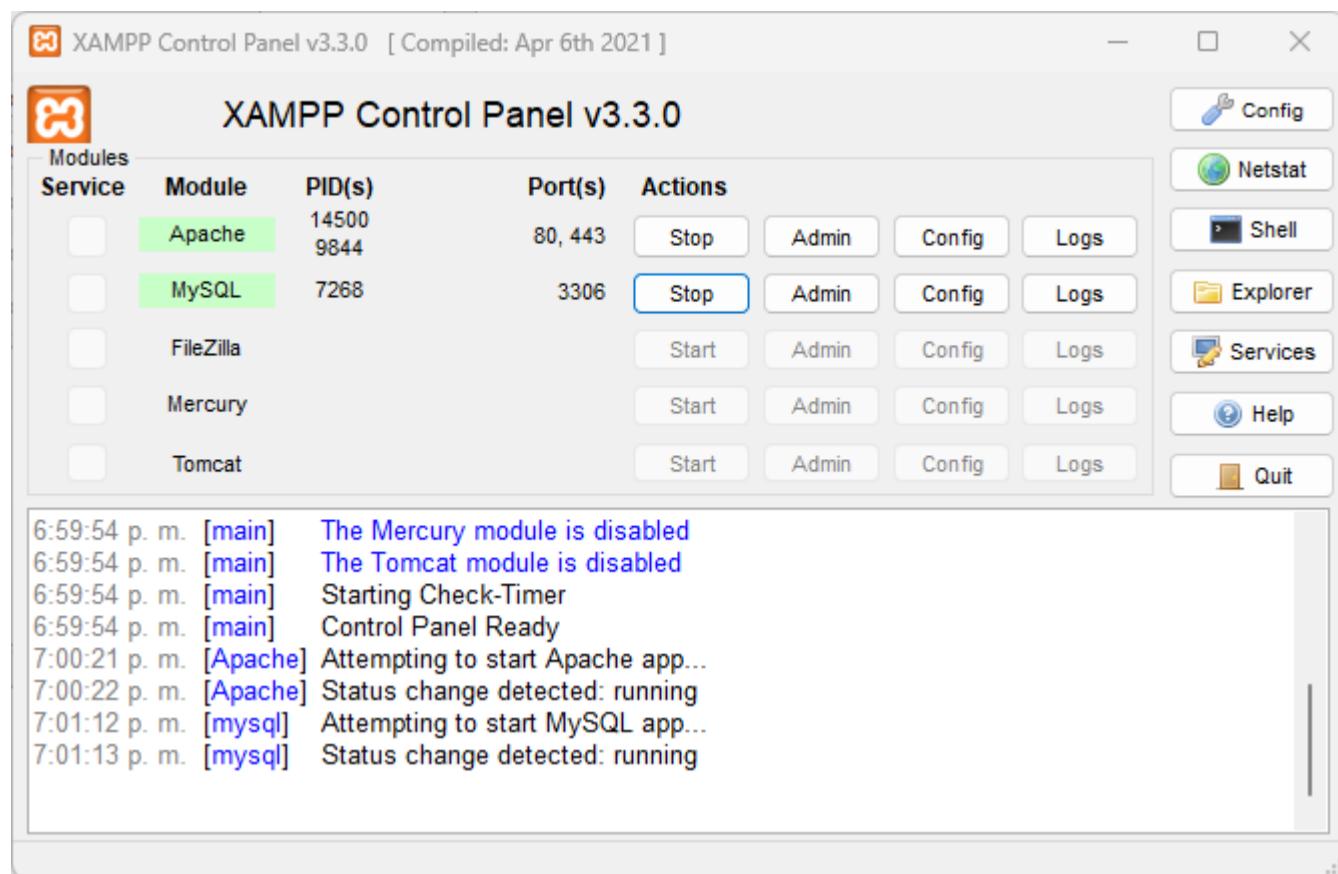


Ilustración 15: Los dos servicios están activos.

Probando que el servidor de aplicaciones funciona bien

Para saber si el servidor de aplicaciones funciona correctamente, se abre un navegador y en <http://localhost>, debería verse esta pantalla:



Ilustración 16: ¿Apache funciona? abre un navegador y se escribe <http://localhost>.

Iniciando con JavaScript

Para escribir código fuente se ejecuta Notepad++:

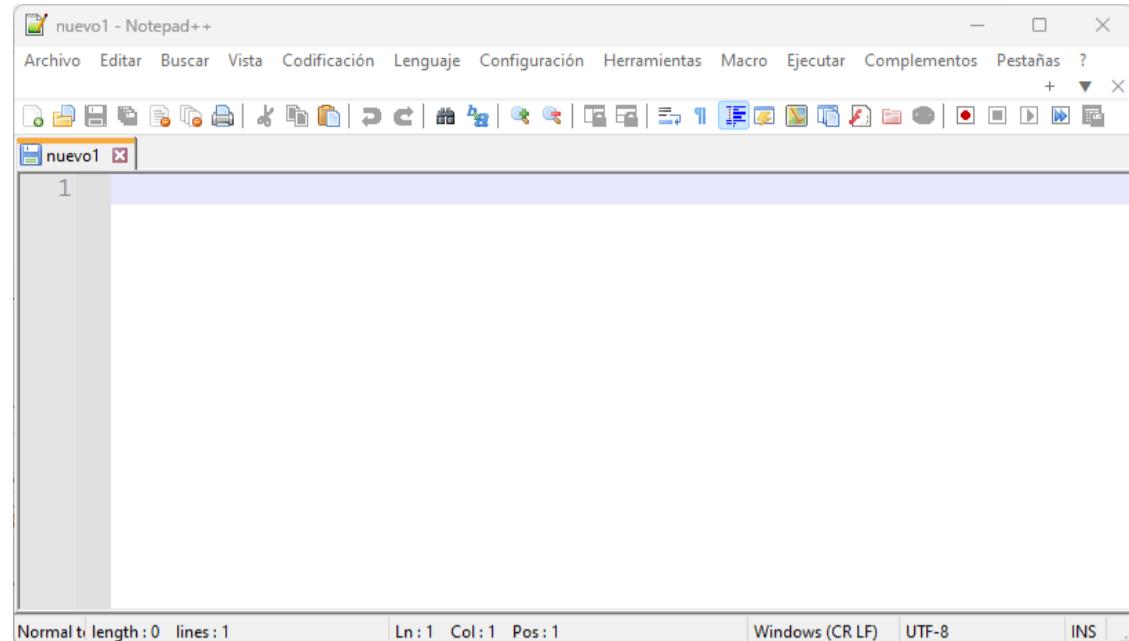


Ilustración 17: Notepad++ en español

El siguiente paso es configurar el Notepad++ para trabajar con JavaScript, para eso se dirige por la opción Lenguaje → H → HTML

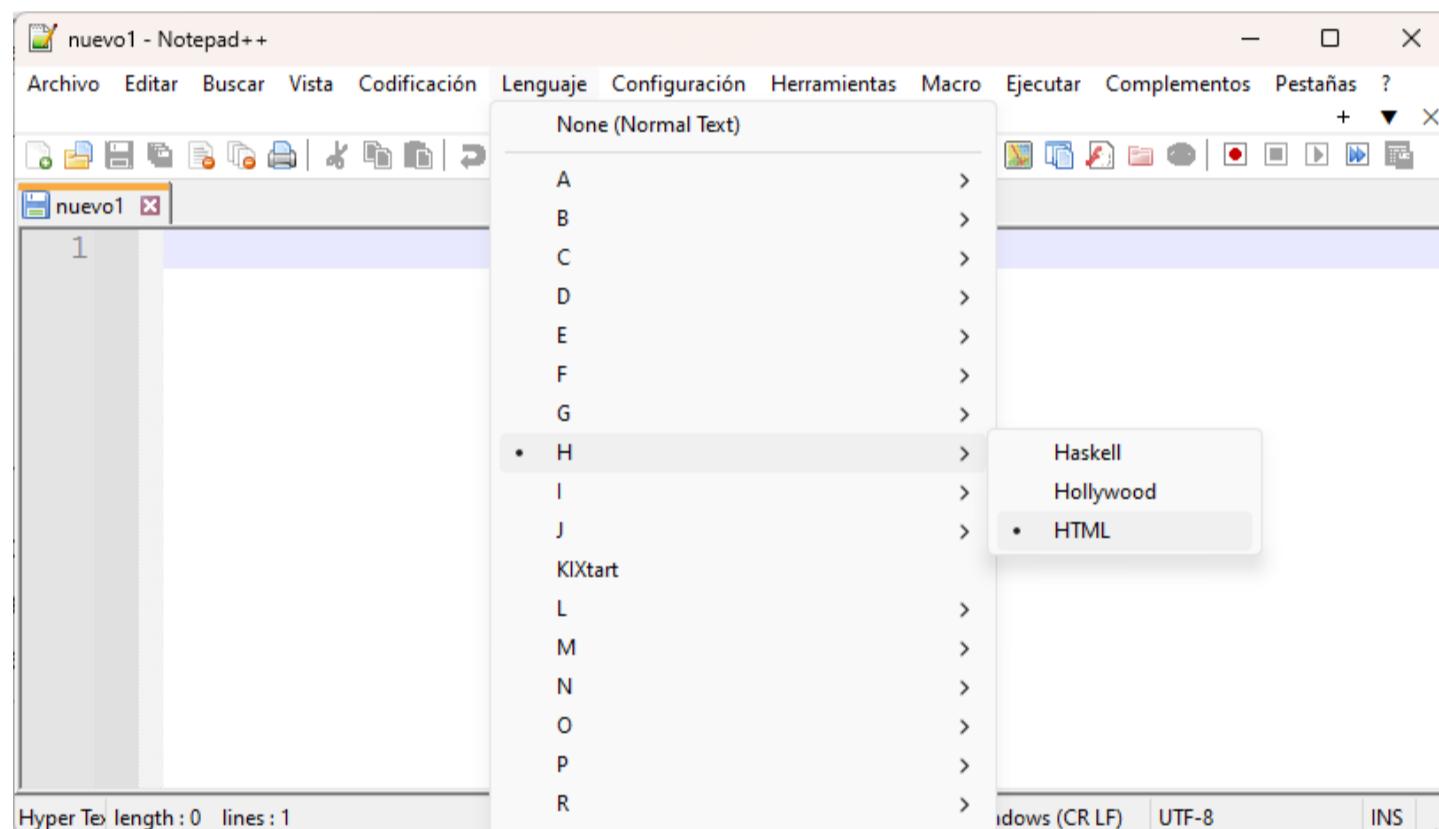


Ilustración 18: Configuramos Notepad++ para editar archivos HTML y JavaScript

¿Por qué por HTML y no JavaScript que está un poco más abajo en el menú? Los 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, habrá una separación del código HTML al de JavaScript (en lecciones más avanzadas).

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

001.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>

</script>
</body>
</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 es posible hacer mantenimiento y software que hay que desechar. 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:

002.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Comentario de una sola línea

    /* Comentarios en
       varias
       líneas */

</script>
</body>
</html>
```

El tradicional “Hola Mundo”

La instrucción document.write es para mostrar mensajes en el navegador. Las instrucciones deben terminar en punto y coma (;)

Nota: En JavaScript, el punto y coma (;) no es estrictamente necesario al final de cada línea porque el lenguaje tiene una característica llamada Inserción Automática de Punto y Coma (Automatic Semicolon Insertion, o ASI). Esto significa que el intérprete intentará adivinar dónde deberían ir los ; y los insertará si son omitidos. Sin embargo, este sistema **no es infalible**.

El documento es guardado como un archivo HTML

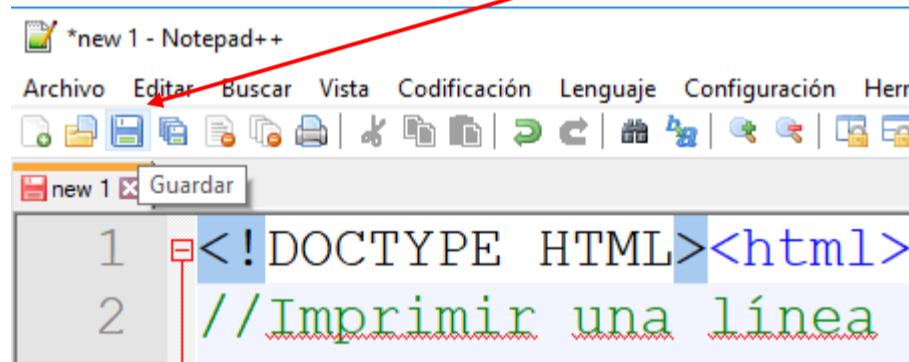


Ilustración 19: Botón de guardar

¿Dónde guardar los archivos HTML?

Una vez instalado XAMPP y que esté ejecutando correctamente, se debe ir a **c:\xampp\htdocs** (o **d:\xampp\htdocs**) y crear una carpeta llamada **src**, en el interior de esa carpeta se ponen todos los archivos que descargó de GitHub.

Nota: Es muy recomendado hacer uso de minúsculas en el nombre de los archivos, también evitar los espacios, tildes y Ñs
¿Por qué? Porque muy seguramente la aplicación WEB futura se aloje en un servidor que tenga el sistema operativo Linux. En Linux se diferencia entre minúsculas y mayúsculas en el nombre de los archivos, el resto de las recomendaciones es para minimizar riesgos con el nombre de los archivos.

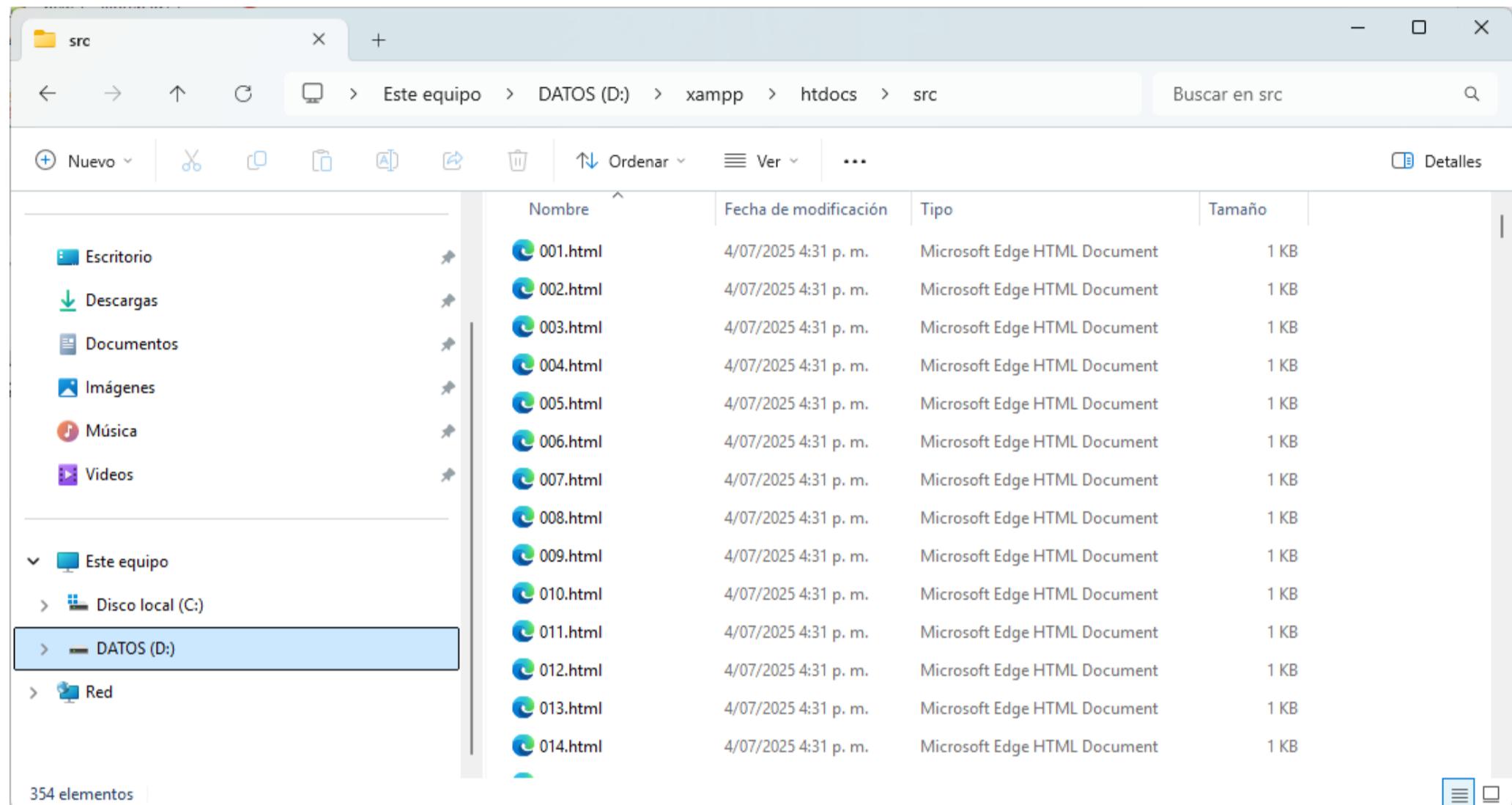


Ilustración 20: En la carpeta XAMPP dentro de HTDOCS se crea SRC

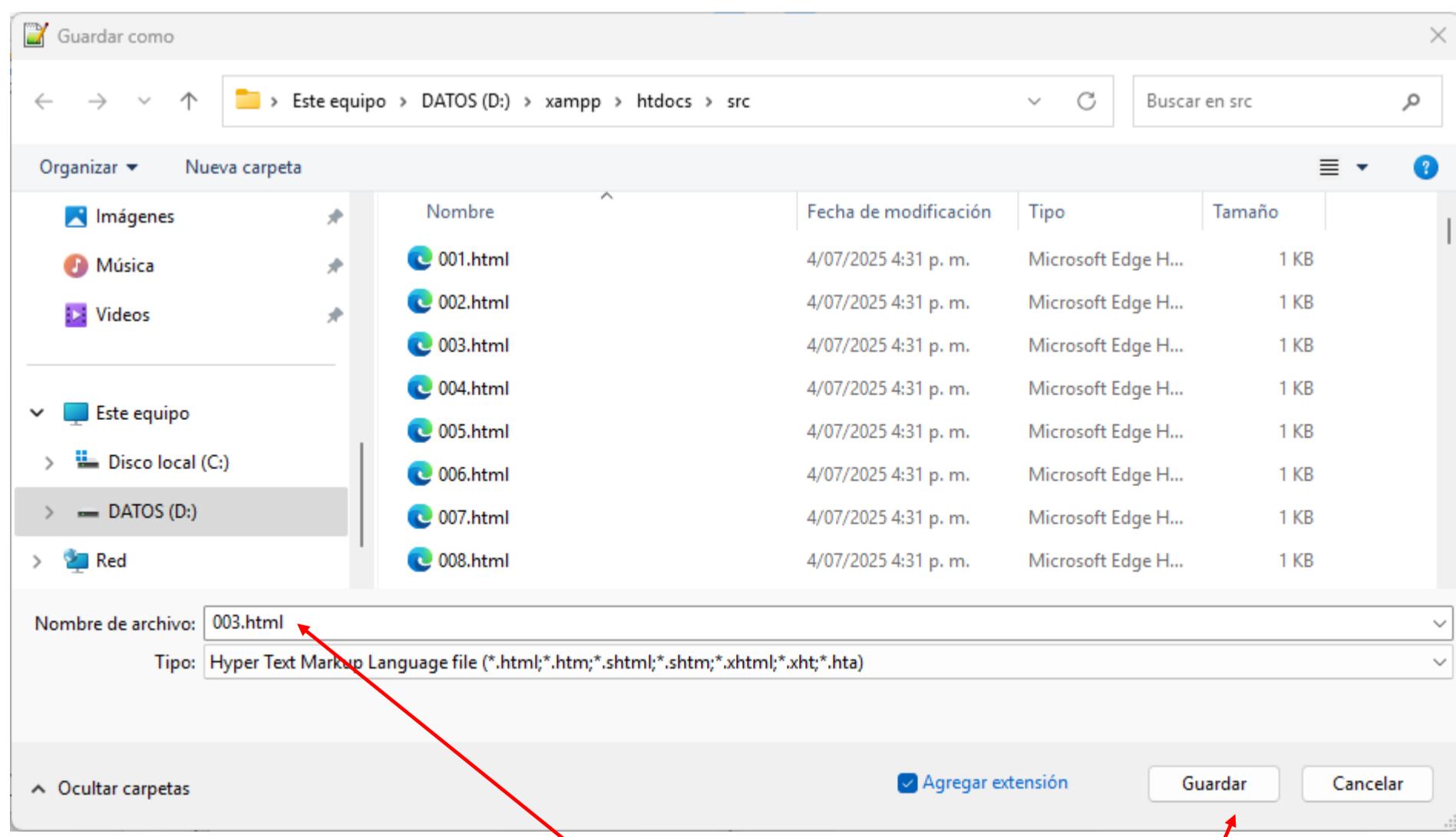
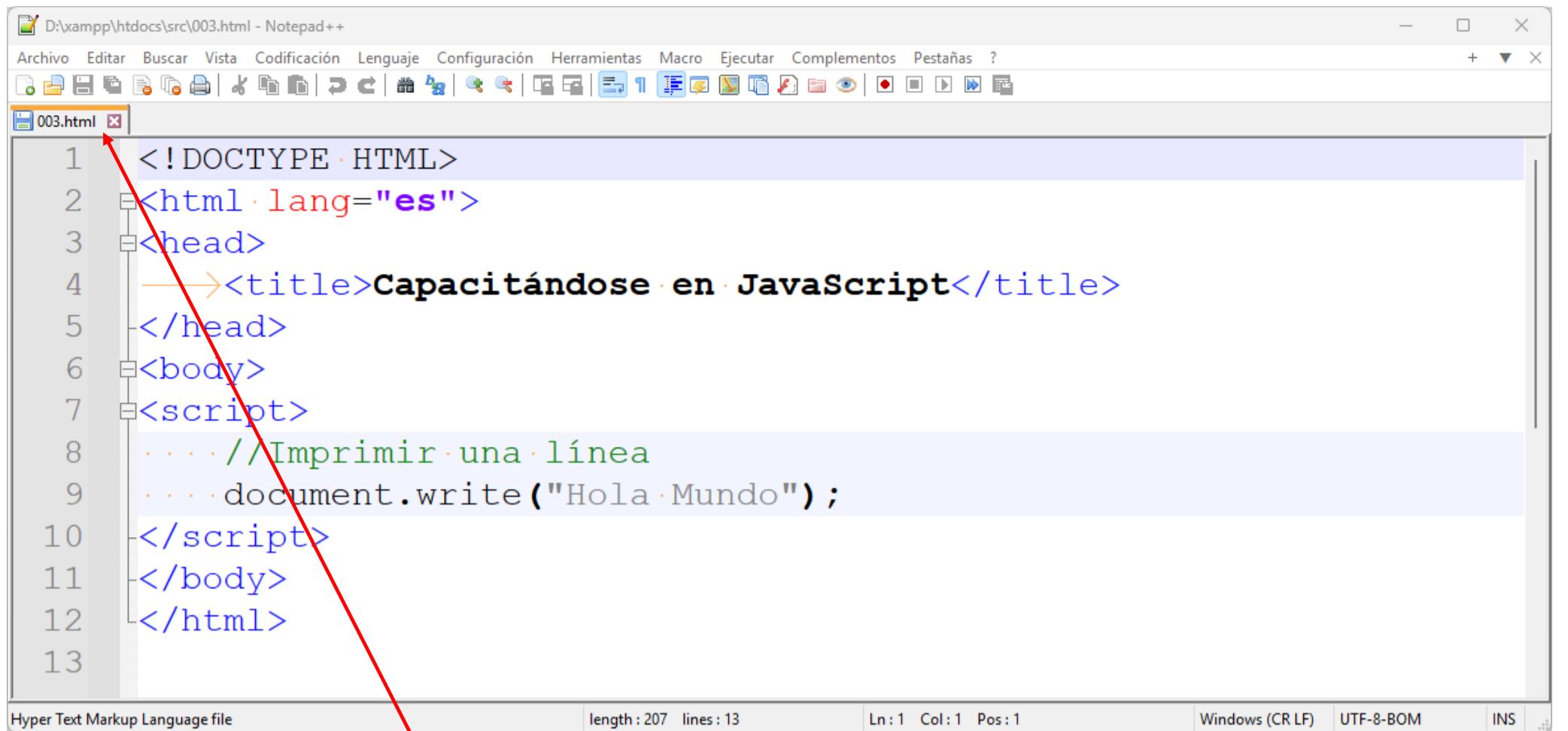


Ilustración 21: Guardar la página HTML en el PC

Hay que estar atento que aparezca "Tipo: Hyper Text Markup Language file" y luego clic en Guardar.



```
1 <!DOCTYPE HTML>
2 <html lang="es">
3 <head>
4     <title>Capacitándose en JavaScript</title>
5 </head>
6 <body>
7 <script>
8     //Imprimir una línea
9     document.write("Hola Mundo");
10 </script>
11 </body>
12 </html>
13
```

Hyper Text Markup Language file | length : 207 lines : 13 | Ln : 1 Col : 1 Pos : 1 | Windows (CR LF) | UTF-8-BOM | INS

Ilustración 22: Hay que fijarse que el archivo tenga extensión .html

Hay que fijarse que la extensión del archivo sea .html

El siguiente paso es ejecutar la página Web, luego debe abrir el navegador y digitar <http://localhost> y luego ir al directorio src, esta sería la pantalla

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
Parent Directory		-	
001.html	2025-07-04 16:31	148	
002.html	2025-07-04 16:31	246	
003.html	2025-07-04 16:31	210	
004.html	2025-07-04 16:31	210	
005.html	2025-07-04 16:31	217	

Ilustración 23: El directorio de los ejemplos de JavaScript

003.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Imprimir una línea
    document.write("Hola Mundo");
</script>
</body>
</html>
```

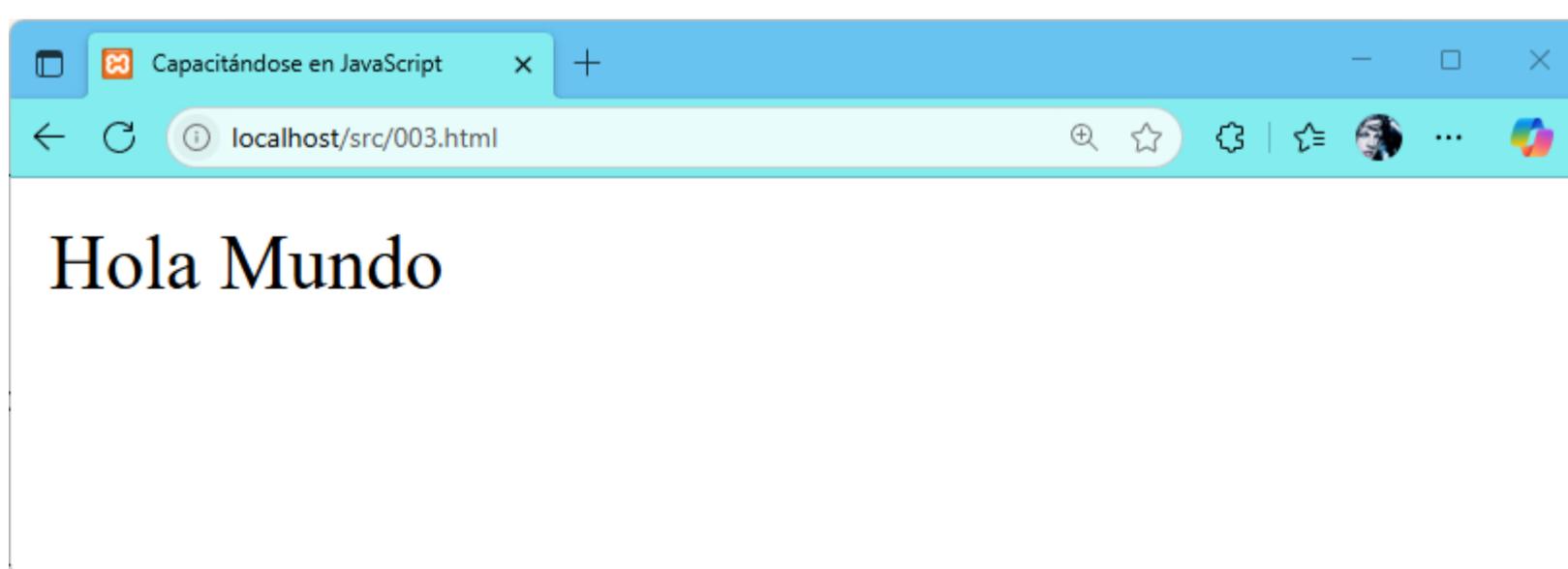


Ilustración 24: Ejecución del programa

Las comillas dobles pueden ser reemplazadas por comillas simples.

004.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Imprimir una linea
    document.write('Hola Mundo');
</script>
</body>
</html>
```

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

005.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Imprimir una linea
    document.write("<b>Hola Mundo</b>");
</script>
</body>
</html>
```

Uso de variables

JavaScript declara las variables con la palabra reservada let. Es un lenguaje débilmente tipado, es decir, no se definen 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).

006.html

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

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

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

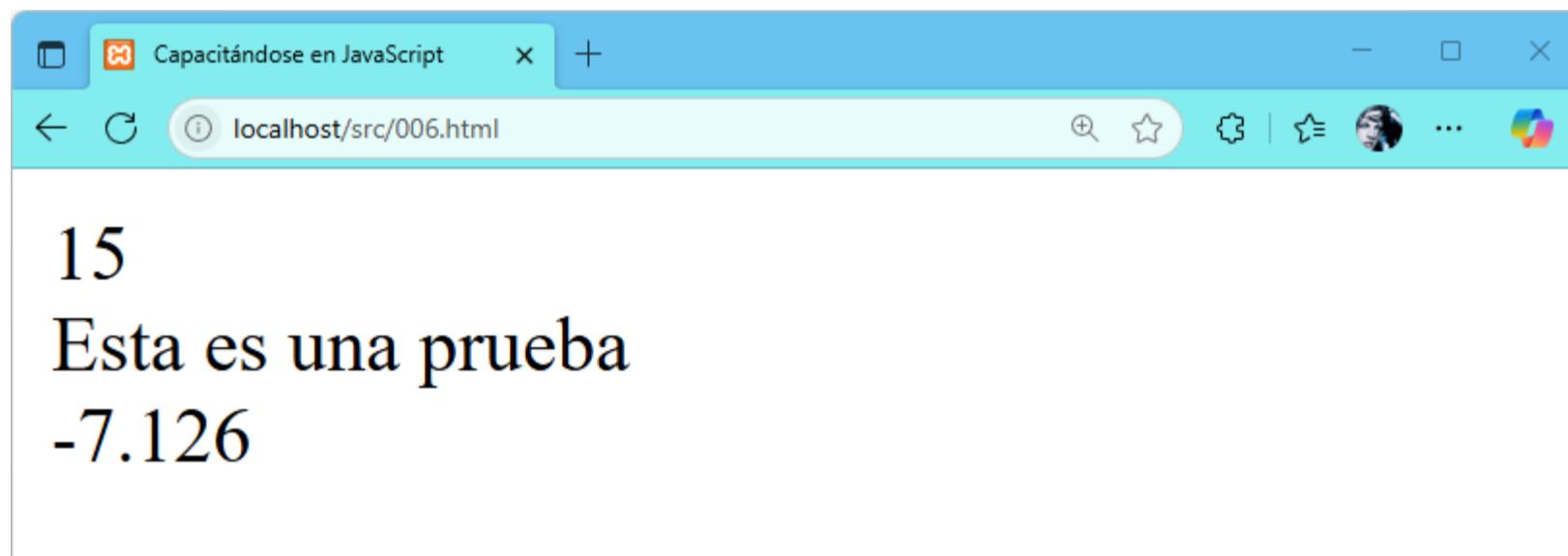


Ilustración 25: Impresión del valor de cada variable

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

007.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Variables en JavaScript
    let valA = 78;
    let cadena = "Cadena de texto";
    let aproxima = -90.23;

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Variables en JavaScript
    let val = -901;
    let cad = "Manual de JavaScript";
    let aprox = 864.5;

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

Operaciones matemáticas

Están disponibles suma(+), resta(-), multiplicación(*), división(/), potencia(**) y división modular(%). La división modular retorna el **residuo** de la división. Este es el código:

009.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Aritmética
    let valA = 12 + 45;
    let valB = 55 - 89;
    let valC = 7 * 8;
    let valD = 21 / 3;
    let valE = 5 ** 3; //5 al cubo
    let 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>
</body>
</html>
```

Asignación

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

010.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Asignación
    let valA = 8;
    let 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>
</body>
</html>
```

Operaciones con cadenas

Se utiliza el operador + para concatenar cadenas.

011.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Uso de cadenas
    let txtA = "Una";
    let txtB = "Cadena";
    let txtC = txtA + " --- " + txtB;
    document.write(txtC);
</script>
</body>
</html>
```

¿Y qué sucede al combinar números y texto?

012.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Uso de cadenas
    let txtA = "Una";
    let txtB = 45.98;
    let txtC = txtA + " --- " + txtB;
    document.write(txtC);
</script>
</body>
</html>
```

Otro cambio, ambas variables con números reales.

013.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Uso de cadenas
    let txtA = 27.6;
    let txtB = 45.98;
    let txtC = txtA + " --- " + txtB;
    document.write(txtC);
</script>
</body>
</html>
```

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

014.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Uso de cadenas
    let txtA = 7;
    let txtB = 5;
    let txtC = txtA + txtB + "aaaaa";
    document.write(txtC);
</script>
</body>
</html>
```

Pasa esto:

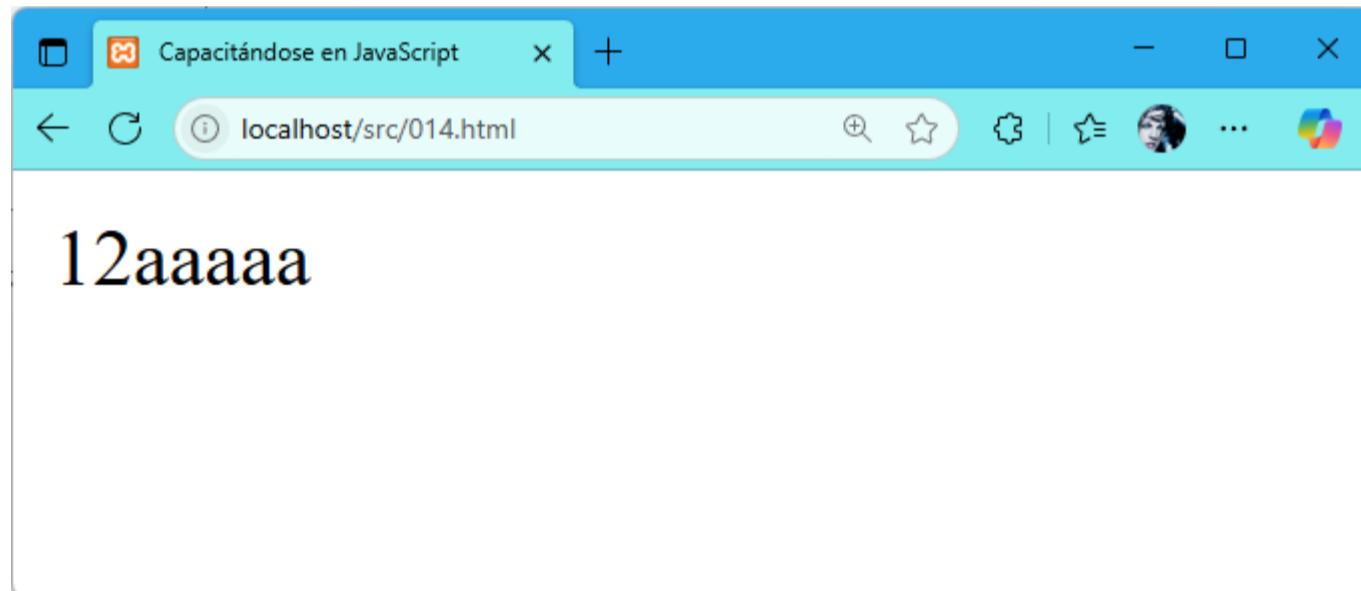


Ilustración 26: Primero se hace la operación matemática y luego la concatenación

¿Y si se hacen más operaciones matemáticas?

015.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Uso de cadenas
    let txtA = 7;
    let txtB = 5;
    let txtC = txtA + txtB + "aaaaa" + txtA + txtB;
    document.write(txtC);
</script>
</body>
</html>
```

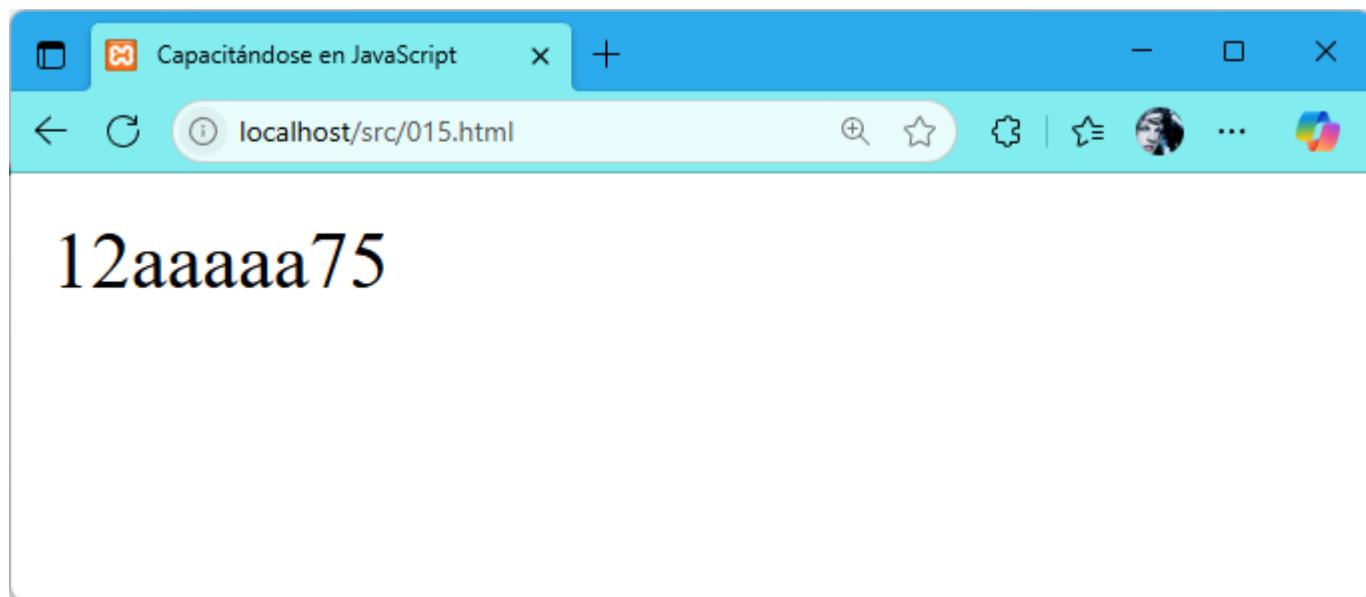


Ilustración 27: Se hace la operación matemática y el resto es concatenación

Impresión de caracteres propios del idioma español

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

C:\tmp\Inicio.html - Notepad++

Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Plugins

Inicio.html

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

Ilustración 28: Imprimir caracteres propios del idioma español

Y al ejecutarlo esto sucede en Google Chrome:

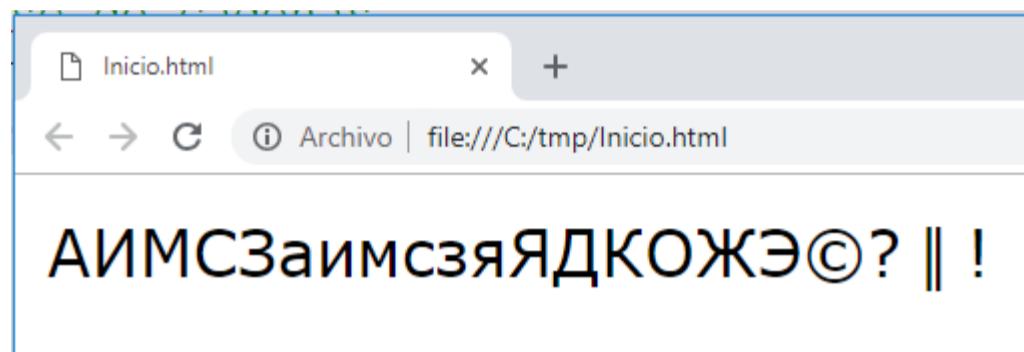


Ilustración 29: En Chrome puede pasar que no se impriman

Funciona correcto en Mozilla Firefox:

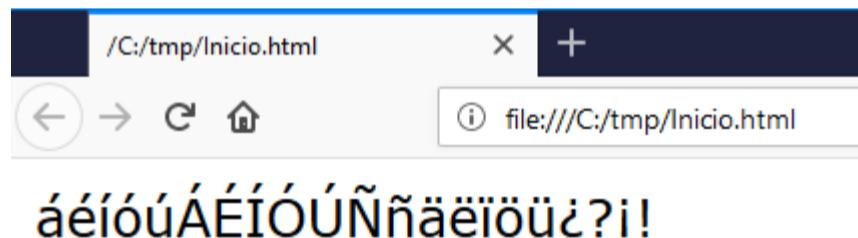


Ilustración 30: Funciona en Mozilla Firefox

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

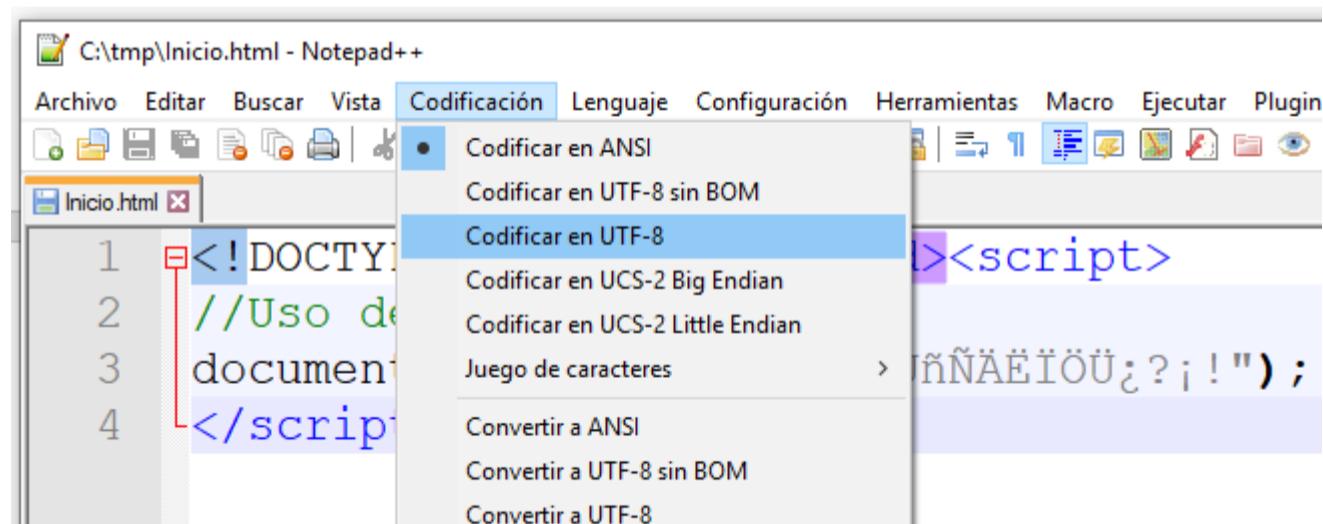


Ilustración 31: Ante un fallo así, verificamos la codificación. Usualmente está en "Codificar en ANSI"

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:

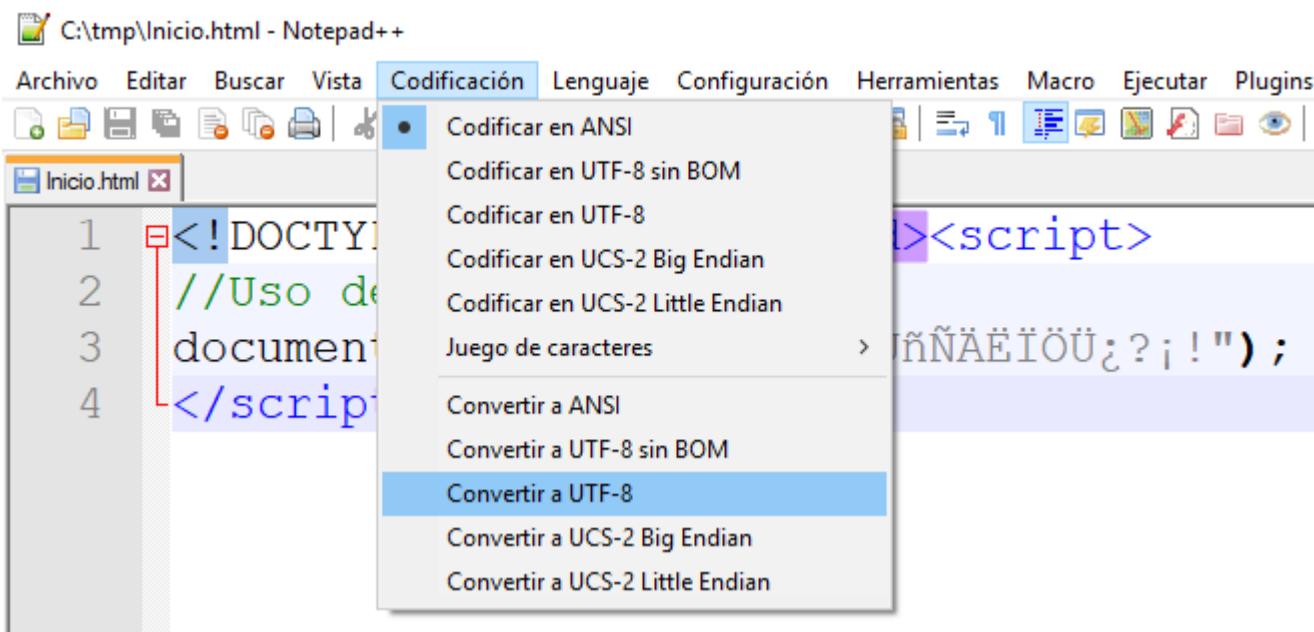


Ilustración 32: Seleccionamos la opción de "Convertir a UTF-8"

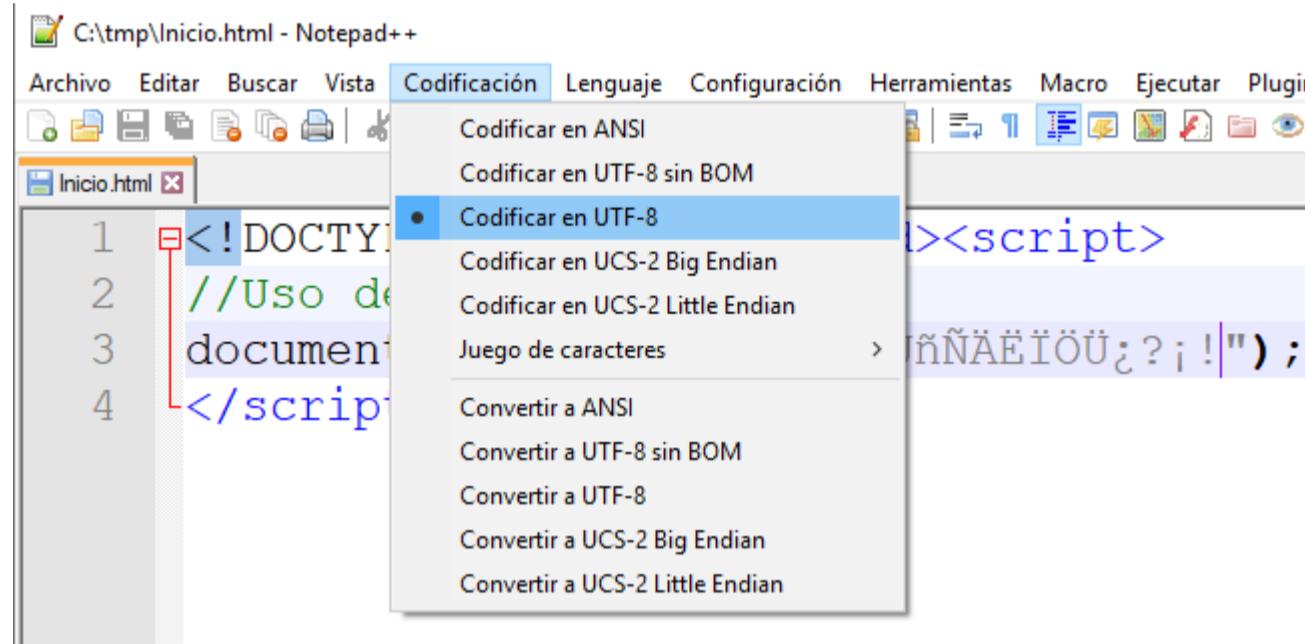


Ilustración 33: Despues de guardar, se chequea que esté en la opción "Codificar en UTF-8"

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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>
</body>
</html>
```

Funciones trigonométricas

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

017.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Librería matemática.
    let angGrados = 35;
    document.write("Grados: " + angGrados);
    let angRadian = angGrados * Math.PI / 180;
    document.write("<br>Radianes: " + angRadian);
    let seno = Math.sin(angRadian); //Seno
    document.write("<br>Seno: " + seno);
    let coseno = Math.cos(angRadian); //Coseno
    document.write("<br>Coseno: " + coseno);
    let tangente = Math.tan(angRadian); //Tangente
    document.write("<br>Tangente: " + tangente);
    let arcoseno = Math.asin(seno); //Arcoseno
    document.write("<br>Arcoseno: " + arcoseno);
    let arcocoseno = Math.acos(coseno); //Arcocoseno
    document.write("<br>Arcocoseno: " + arcocoseno);
    let arcotangente = Math.atan(tangente); //Arcotangente
    document.write("<br>Arcotangente: " + arcotangente);
</script>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Captura de datos por pantalla
    let texto = prompt("Escriba un mensaje");
    document.write("Escribió: " + texto);
</script>
</body>
</html>
```

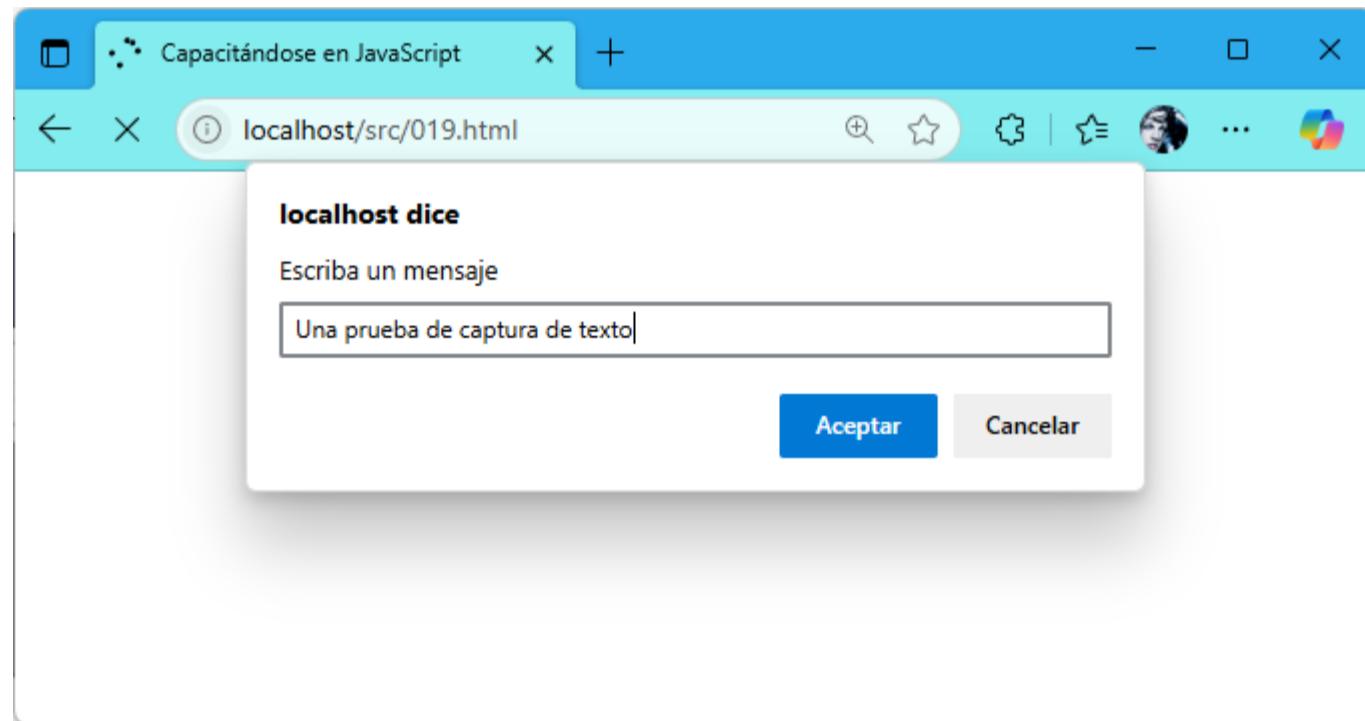


Ilustración 34: Cuadro de diálogo para capturar un valor texto



Ilustración 35: 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.

020.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Captura de datos numéricos
    let numEntero = parseInt(prompt("Escriba un valor entero")); //Tipo entero
    let numReal = parseFloat(prompt("Escriba un valor real")); //Tipo real
    let suma = numEntero + numReal;
    document.write(suma);
</script>
</body>
</html>
```

Ejemplos de cálculos

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

021.html

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

Área y perímetro del círculo

022.html

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

Área y volumen del cilindro

023.html

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

Corrigiendo errores en JavaScript

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

El siguiente código contiene varios errores:

```
C:\Users\engin\OneDrive\Proyecto\Libro\22. JSAlgoritmo>ErrorSintaxis.html - Notepad++
Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Complementos Pestañas ?
ErrorSintaxis.html x Rafael Alberto Moreno Parra (rafael)
1 <!DOCTYPE HTML>
2 <html lang="es">
3 <head>
4     <title>Capacitándose en JavaScript</title>
5 </head>
6 <body>
7 <script>
8     //Área y volumen del cilindro
9     let radio = parseFloat(pronpt("Radio del círculo de la tapa"));
10    let altura = ParseFloat(prompt("Altura del cilindro"));
11    lett area = 2 * Math.PI * radio * (altura + radio);
12    let volumen = Mth.PI * radio * radio * altura;
13    Document.write("Área es: " + area)
14    document.wite("<br>Volumen es: " + volumen);
15 </script>
16 </body>
17 </html>
18
```

Hyper Text Markup Language file length : 515 lines : 18 Ln : 9 Col : 32 Pos : 187 Windows (CR LF) UTF-8-BOM INS

Ilustración 36: Código en JavaScript con varios errores

Al ejecutar en Edge, la pantalla aparece en blanco

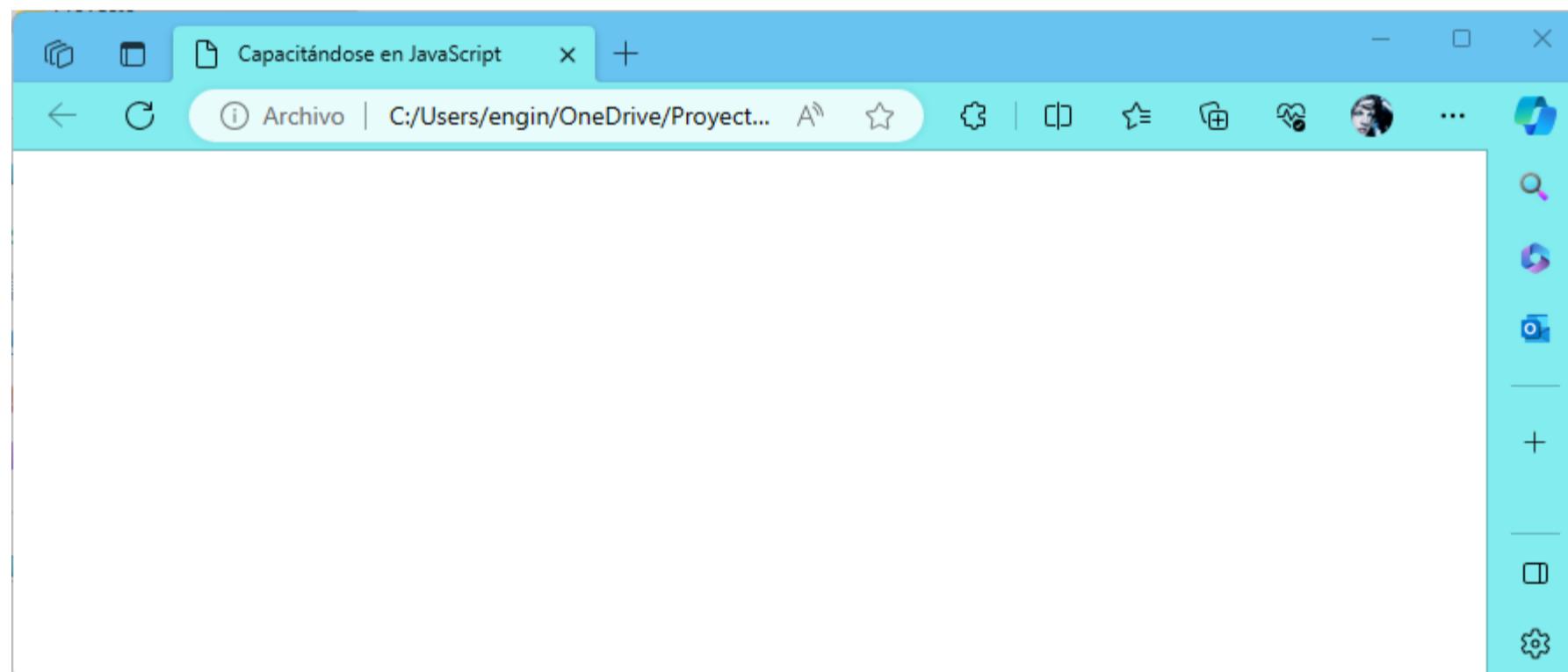


Ilustración 37: Ante errores en JavaScript, el navegador queda en blanco

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

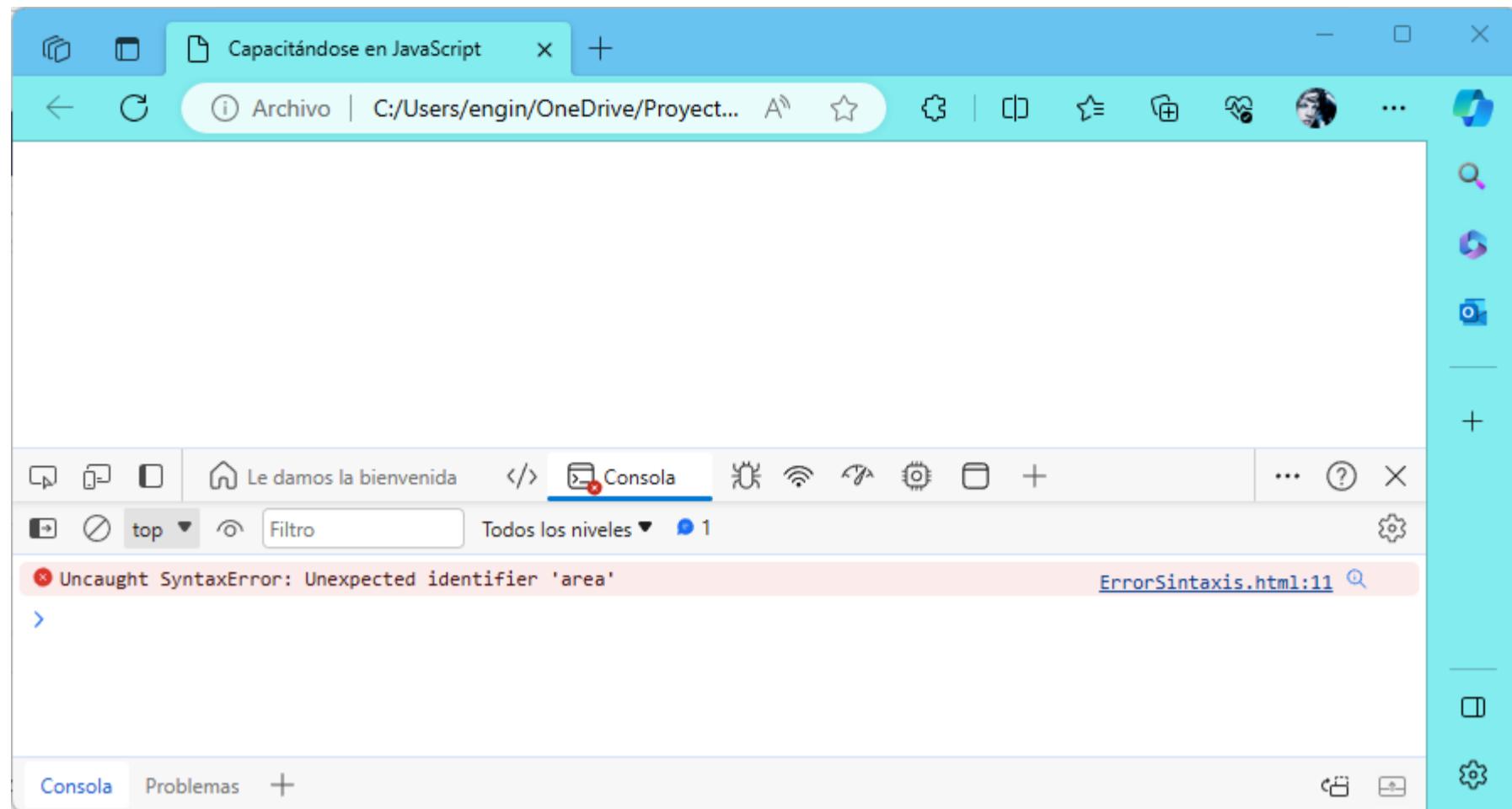


Ilustración 38: Presionando F12 en el navegador Edge y dando clic en Consola se muestra el error

En “Consola” se muestra en qué línea hay un error. Se debe entender el error y proceder a corregirlo. No es esperable una identificación perfecta del error por parte del navegador. En otros navegadores es similar el uso de F12 para mostrar la consola.

Validación en línea del código

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

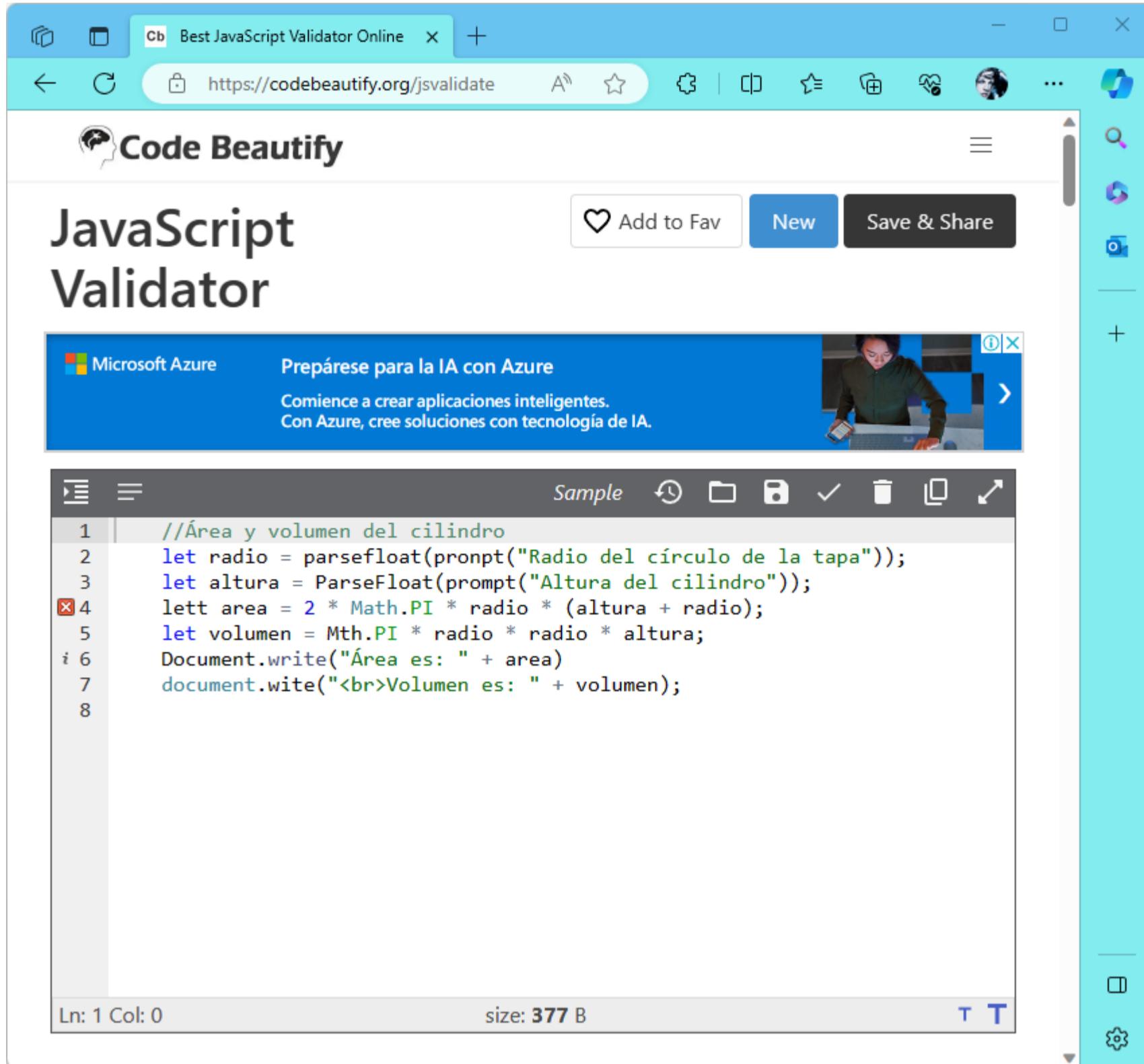


Ilustración 39: JavaScript Validator para chequear sintaxis

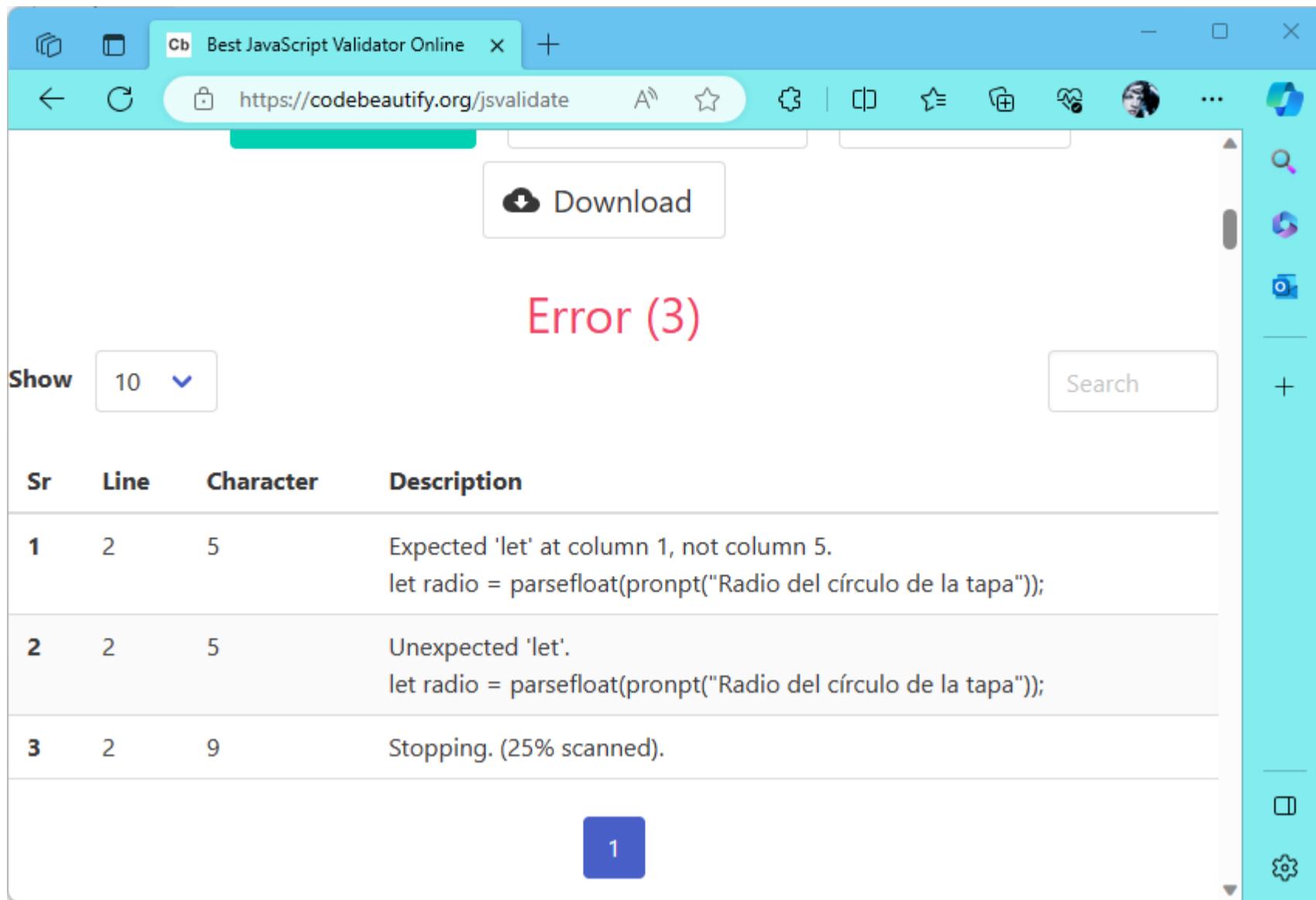


Ilustración 40: Diagnóstico de JavaScript Validator de un script de ejemplo

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

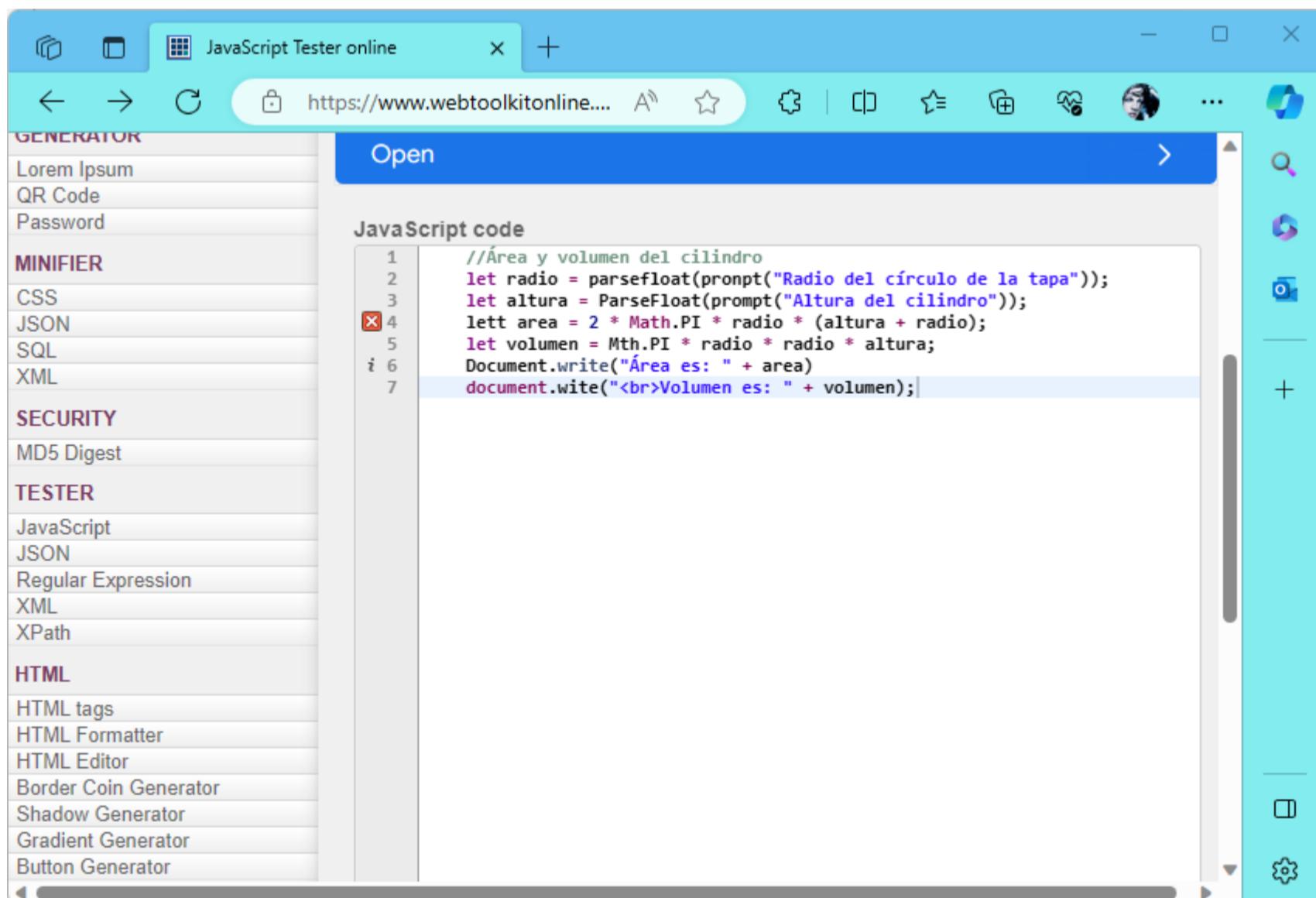


Ilustración 41:JavaScript Tester para validar sintaxis

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

The screenshot shows a browser window with the URL <https://jshint.com/>. The page displays a block of JavaScript code and its analysis results. On the left, the code is shown:

```
//Área y volumen del cilindro
let radio = parseFloat(prompt("Radio del círculo de la tapa"));
let altura = ParseFloat(prompt("Altura del cilindro"));
lett area = 2 * Math.PI * radio * (altura + radio);
let volumen = Mth.PI * radio * radio * altura;
Document.write("Área es: " + area)
document.wite("Volumen es: " + volumen);
```

On the right, the results are presented in sections:

- CONFIGURE**
- Six warnings**
 - 2 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
 - 3 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
 - 4 Expected an assignment or function call and instead saw an expression.
 - 4 Missing semicolon.
 - 5 'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
 - 6 Missing semicolon.
- Six undefined variables**
 - 2 parseFloat
 - 2 prompt
 - 3 ParseFloat
 - 4 lett
 - 4 area
 - 5 -----

The sidebar on the right contains the JSHint logo and version information (version 2.13.6), along with links to About, Documentation, Install, Contribute, and Blog.

Ilustración 42: Jshint para validar sintaxis

Sí condicional

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

024.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Si condicional
    let valA = parseFloat(prompt("valor A: "));
    let 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>
</body>
</html>
```

025.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Si condicional. Uso de else if
    let valA = parseFloat(prompt("valor A: "));
    let 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>
</body>
</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 ||

026.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    let ladoA = parseInt(prompt("Lado A:"));
    let ladoB = parseInt(prompt("Lado B:"));
    let 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>
</body>
</html>
```

027.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Tipo de triángulo
    let ladoA = parseInt(prompt("Lado A:"));
    let ladoB = parseInt(prompt("Lado B:"));
    let 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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Uso del switch
    let 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>
</body>
</html>
```

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Uso del switch
    let 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>
</body>
</html>
```

Ciclos. Uso del “for”

030.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ciclo for creciente
    for (let cont = 1; cont <= 100; cont++) {
        document.write(cont + "<br>");
    }
</script>
</body>
</html>
```

031.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ciclo for decreciente
    for (let cont = 100; cont >= 1; cont--) {
        document.write(cont + "<br>");
    }
</script>
</body>
</html>
```

032.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ciclo for creciente de 3 en 3
    for (let cont = 0; cont <= 300; cont += 3) {
        document.write(cont + "<br>");
    }
</script>
</body>
</html>
```

033.html

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

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

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Ciclo for con dos variables. Observemos las dos
       condiciones que deben darse */
    for (let 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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Ciclo for con dos variables. Observemos las dos
       condiciones que deben darse cambiando && por || */
    for (let 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>
</body>
</html>
```

Ciclo while

038.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ciclo while creciente
    let cont = 1;
    while (cont <= 100) {
        document.write(cont + "<br>");
        cont++;
    }
</script>
</body>
</html>
```

039.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ciclo while decreciente
    let cont = 100;
    while (cont >= 1) {
        document.write(cont + "<br>");
        cont--;
    }
</script>
</body>
</html>
```

040.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ciclo while creciente de 3 en 3
    let cont = 0;
    while (cont <= 300) {
        document.write(cont + "<br>");
        cont += 3;
    }
</script>
</body>
</html>
```

041.html

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ciclo while creciente aumentando el doble del anterior
    let 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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ciclo while decreciente disminuyendo la mitad del anterior
    let 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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Ciclo while con dos variables. Observemos las dos
       condiciones que deben darse */
    let x = 1;
    let 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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Ciclo while con dos variables. Observemos las dos
       condiciones que deben darse cambiando && por || */
    let x = 1;
    let 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>
</body>
</html>
```

Ciclo do - while

046.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Ciclo do-while con dos variables. Observemos las dos
       condiciones que deben darse cambiando && por || */
    let x = 1;
    let 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>
</body>
</html>
```

047.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ciclo do-while creciente
    let cont = 1;
    do {
        document.write(cont + "<br>");
        cont++;
    } while (cont <= 100);
</script>
</body>
</html>
```

048.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ciclo do-while decreciente
    let cont = 100;
    do {
        document.write(cont + "<br>");
        cont--;
    } while (cont >= 1);
</script>
</body>
</html>
```

049.html

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ciclo do-while creciente de 3 en 3
    let cont = 0;
    do {
        document.write(cont + "<br>");
        cont += 3;
    } while (cont <= 300);
</script>
</body>
</html>

```

050.html

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ciclo do-while creciente de 10 en 10. Se detiene antes de 500
    let cont = 0;
    do {
        document.write(cont + "<br>");
        cont += 10;
    } while (cont < 500);
</script>
</body>
</html>

```

051.html

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ciclo while creciente aumentando el doble del anterior
    let 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>
</body>
</html>

```

052.html

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ciclo while decreciente disminuyendo la mitad del anterior
    let 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>
</body>
</html>

```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Ciclo while con dos variables. Observemos las dos
       condiciones que deben darse */
    let x = 1;
    let 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>
</body>
</html>
```

Uso del break

054.html

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

055.html

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

056.html

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

Uso del continue

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

057.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ciclo for creciente, se salta los pares
    for (let 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>
</body>
</html>
```

058.html

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

059.html

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

Ciclos anidados y salir de estos

060.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ciclos anidados
    for (let x = 1; x <= 3; x++) {
        for (let y = 45; y <= 50; y++) {
            for (let z = 7; z <= 9; z++) {
                document.write(x + ";" + y + ";" + z + "<br>");
            }
        }
    }
</script>
</body>
</html>
```

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

061.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Saliendo de ciclos anidados. Uso de etiquetas.
    saliendo:
    for (let x = 1; x <= 3; x++) {
        for (let y = 45; y <= 50; y++) {
            for (let 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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Saliendo de ciclos anidados
    for (let x = 1; x <= 3; x++) {
        ciclointerno:
        for (let y = 45; y <= 50; y++) {
            for (let 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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    let numEntero = parseInt(prompt("Escriba un valor entero")); //Tipo entero
    let 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 (let divide = 3; divide <= Math.sqrt(num); divide += 2)
            if (num % divide === 0) return false;
        return true;
    }
</script>
</body>
</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

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

Libreria.js

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

//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) {
    let acumula = 0;
    while (num !== 0) {
        if (num % 10 === cifra) acumula++;
        num = Math.floor(num / 10);
    }
    return acumula;
}

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

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

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

//Retorna la sumatoria de las cifras de un número
function SumaCifras(num) {
    let acum = 0;
    while (num !== 0) {
        let 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) {
```

```

let acum = 0;
while (num !== 0) {
    let 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) {
    let acum = 0;
    while (num !== 0) {
        let 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) {
    let acum = 1;
    while (num !== 0) {
        let 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) {
    let acum = 1;
    while (num !== 0) {
        let 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) {
    let acum = 1;
    while (num !== 0) {
        let 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 (let 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) {
    let 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) {
        let 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) {
    let acum = 0;
    while (num !== 0) {
        let cifra = num % 10;
        if (cifra % 2 === 0) acum++;
        num = Math.floor(num / 10);
    }
    return acum;
}

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

//Retorna la cifra más alta
function LaCifraMasAlta(num) {
    let 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) {
    let 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 (let cifra = 0; cifra <= 9; cifra++) {
        let numero = num;
        let 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 (let cifra = 0; cifra <= 8; cifra += 2) {
        let numero = num;
        let 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 (let cifra = 1; cifra <= 9; cifra += 2) {
        let numero = num;
        let 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) {
    let 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>

064.html

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

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <script src="Libreria.js"></script>
</head>
<body>
<script>
    let minimo = 1000;
    let maximo = 9999;

    //Cantidad de números que la suma de las tres últimas cifras es par
    let cuenta = 0;
    for (let 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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <script src="Libreria.js"></script>
</head>
<body>
<script>
    let minimo = 1000;
    let maximo = 9999;

    //Cantidad de números que la multiplicación de las tres últimas cifras es igual a la suma de esas tres
    //últimas cifras
    let cuenta = 0;
    for (let 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>
</body>
</html>
```

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <script src="Libreria.js"></script>
</head>
<body>
<script>
    let minimo = 8000;
    let maximo = 9000;

    //Cantidad de números que cada una de sus tres últimas cifras es menor de 5
    let cuenta = 0;
    for (let 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>
</body>
</html>

```

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <script src="Libreria.js"></script>
</head>
<body>
<script>
    let minimo = 8500;
    let maximo = 8700;

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

```

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <script src="Libreria.js"></script>
</head>
<body>
<script>
    let minimo = 4000;
    let maximo = 7000;

    //Cantidad de números primos que no tengan el número 3 en sus cifras
    let contar = 0;
    for (let 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>
</body>
</html>

```

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <script src="Libreria.js"></script>
</head>
<body>
<script>
    let minimo = 1000;
    let maximo = 9999;

    //Cantidad de números palíndromos que tengan mínimo dos veces el 7 en sus cifras
    let contar = 0;
    for (let 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>
</body>
</html>

```

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <script src="Libreria.js"></script>
</head>
<body>
<script>
    let minimo = 5000;
    let maximo = 9000;

    //Cantidad de números que tengan solo cifras pares y al menos un número 4
    let contar = 0;
    for (let 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>
</body>
</html>

```

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <script src="Libreria.js"></script>
</head>
<body>
<script>
    let minimo = 7700;
    let maximo = 8000;

    //Cantidad de números que la suma de sus cifras es impar y al menos tenga una vez el número 7
    let contar = 0;
    for (let 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>
</body>
</html>

```

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <script src="Libreria.js"></script>
</head>
<body>
<script>
    let minimo = 1000;
    let maximo = 9999;

    //Cantidad de números primos que sus cifras están entre 3 y 7
    let contar = 0;
    for (let 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>
</body>
</html>

```

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <script src="Libreria.js"></script>
</head>
<body>
<script>
    let minimo = 10;
    let maximo = 999999;

    //Cantidad de números primos y a su vez palíndromos
    let contar = 0;
    for (let 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>
</body>
</html>

```

Método de Bisección

Ver: https://es.wikipedia.org/wiki/M%C3%A9todo_de_bisecci%C3%B3n

075.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Algoritmo de bisección
    let Xinitial = 1;
    let Xfinal = 2;
    let Ciclos = 100; //Número de veces que hará la operación de aproximación

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

    //Retorna el punto X entre Xinitial y Xfinal que más se aproxime al punto de corte
    function Biseccion(Xinitial, Xfinal, Ciclos) {
        let Xmedio = 0;
        if (ecuacion(Xinitial) === 0) Xmedio = Xinitial;
        else if (ecuacion(Xfinal) === 0) Xmedio = Xfinal;
        else
            for (let cont = 1; cont <= Ciclos; cont++) {
                Xmedio = (Xinitial + Xfinal) / 2;
                if (ecuacion(Xmedio) === 0) break;
                else if (ecuacion(Xinitial) * ecuacion(Xmedio) < 0) Xfinal = Xmedio;
                else Xinitial = Xmedio;
            }
        return Xmedio;
    }

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ámbito (scope) de una variable
    let 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>
</body>
</html>
```

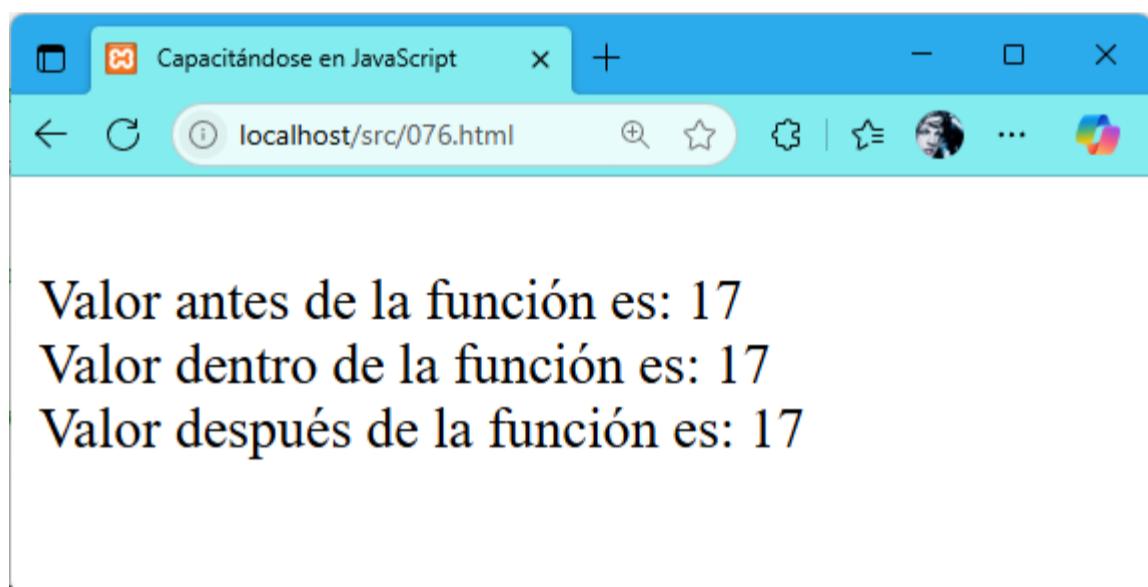


Ilustración 43: Ámbito (scope) de una variable

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    let 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>
</body>
</html>

```

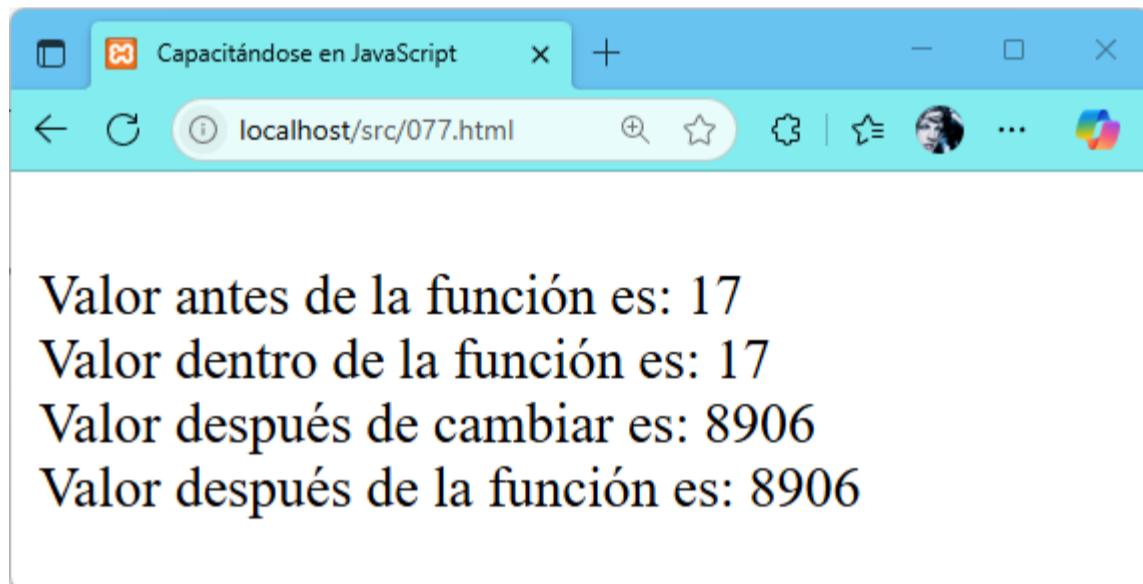


Ilustración 44: Ámbito (scope) de una variable

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    let 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() {
        //Genera mensaje de error
        document.write("<br>Valor dentro de la función es: " + valor);

        let valor = 8906;

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

```

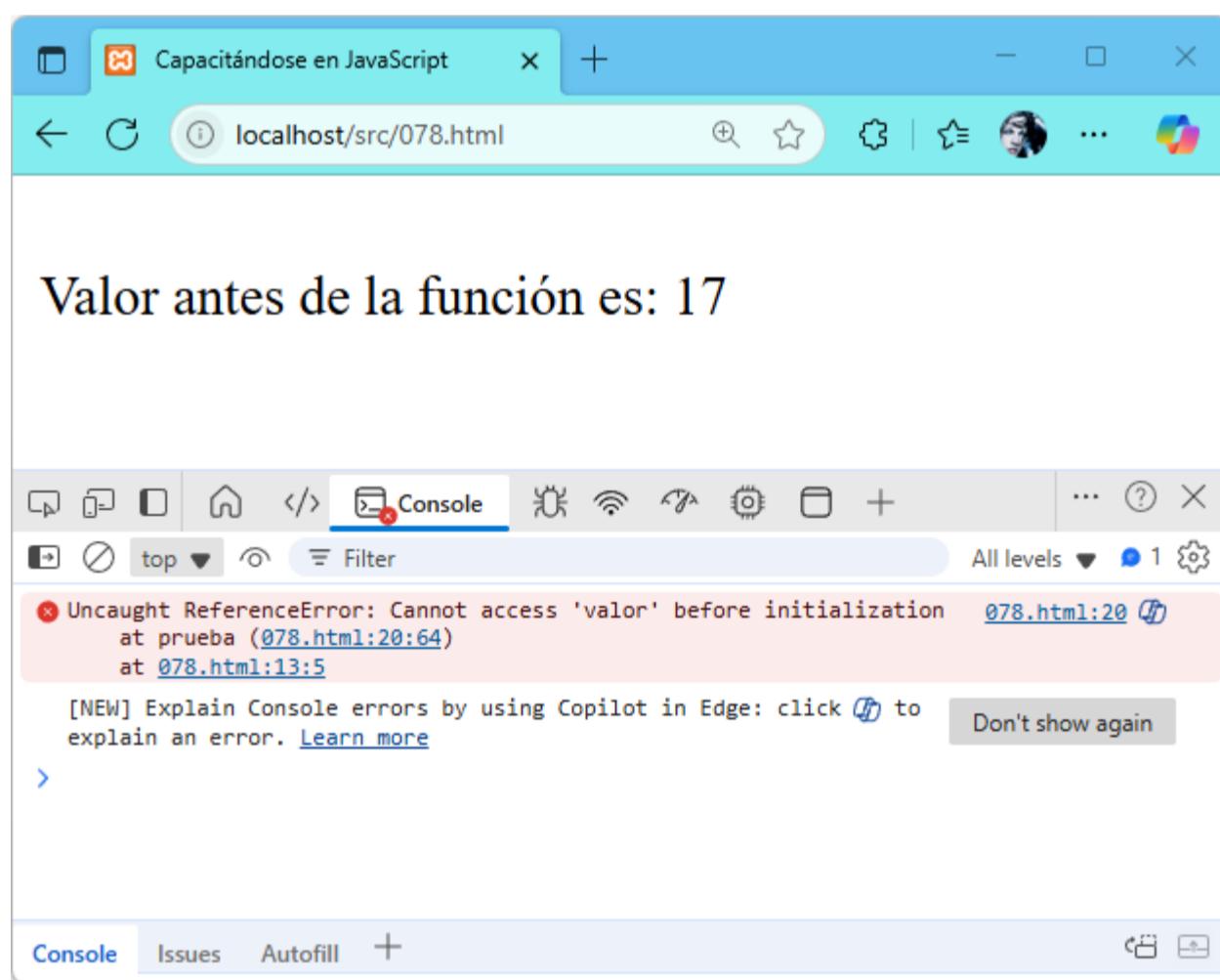


Ilustración 45: Ámbito (scope) de una variable

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ámbito (scope) de una variable
    let 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);

        valor = 8906;

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

```

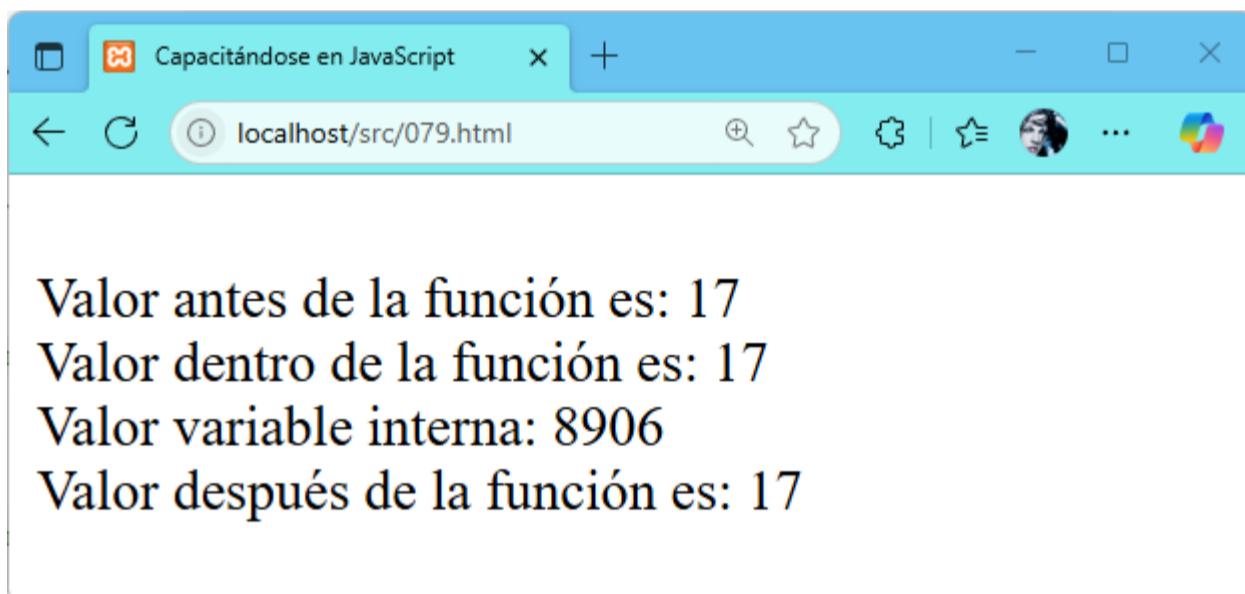


Ilustración 46: Ámbito (scope) de una variable

Funciones recursivas

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

080.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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>
</body>
</html>
```

081.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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);
    }

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Convertir a binario
    ConvierteBinario(70);

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Potencia de un número
    let 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>
</body>
</html>
```

Uso de eval como evaluador de expresiones algebraicas

085.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Uso de eval como evaluador de expresiones algebraicas
    let valor = eval("3+2*5");
    document.write(valor);
</script>
</body>
</html>
```

086.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Uso de eval como evaluador de expresiones algebraicas
    let valor = eval("Math.sin(15)-Math.cos(21)");
    document.write(valor);
</script>
</body>
</html>
```

087.html

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Números aleatorios
    let numero = Math.random(); //Un número entre 0 y 1. Nunca dará 1.
    document.write(numero);
</script>
</body>
</html>
```

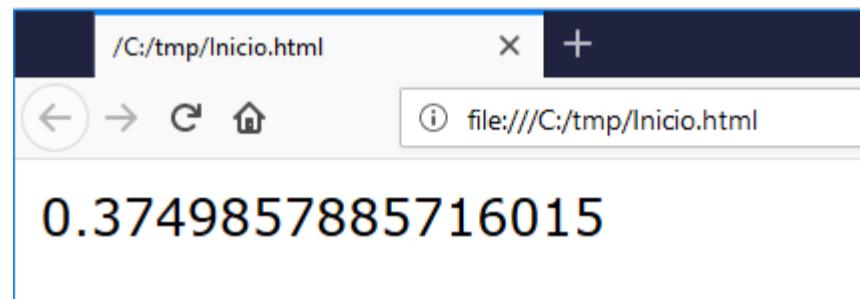


Ilustración 47: Número aleatorio

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Números aleatorios. Ciclo.
    for (let num = 1; num <= 100; num++)
        document.write(Math.random() + ", ");
</script>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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 */
let M = 100;
let D = 7;
for (let cont = 1; cont <= 100; cont++) {
    let r1 = Math.random();
    let r2 = Math.random();
    let c = Math.cos(2 * Math.PI * r2) * Math.sqrt(-2 * Math.log(r1));
    let variable = M + D * c;
    document.write(variable + ", ");
}
</script>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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 =  $A + \sqrt{r * (C-A) * (B-A)}$ 
       de lo contrario
           Variable =  $C - \sqrt{(1-r) * (C-A) * (C-B)}$ 
    */
    let A = 150;
    let B = 190;
    let C = 230;
    let variable = 0;
    for (let cont = 1; cont <= 100; cont++) {
        let 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>
</body>
</html>
```

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

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>

093.html

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

094.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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>
</body>
</html>
```

095.html

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

isFinite

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

096.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Verificar si un número es legal o no
    let dividendo = parseInt(prompt("Digite dividendo"));
    let divisor = parseInt(prompt("Digite divisor"));
    let 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>
</body>
</html>
```

Try...catch

Captura el error matemático

097.html

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

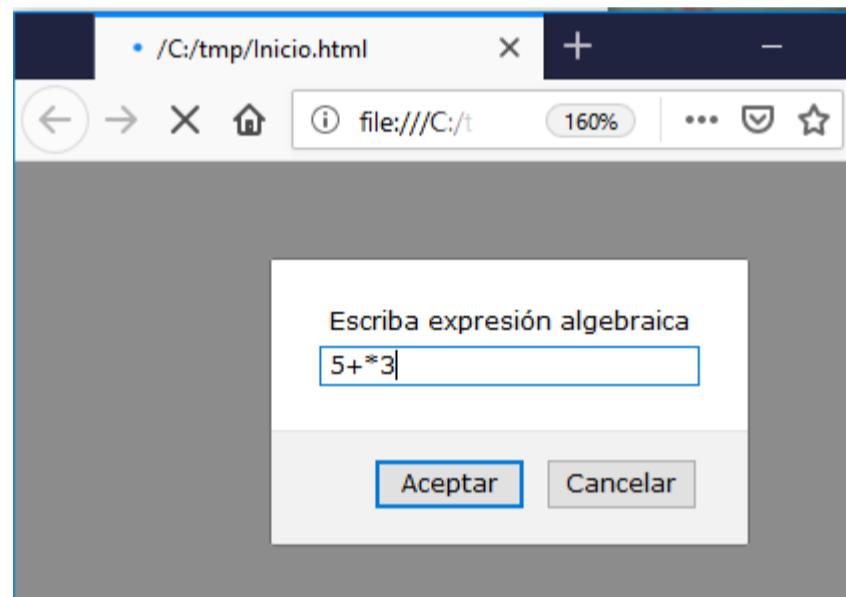


Ilustración 48: Ingresa expresión con sintaxis errónea

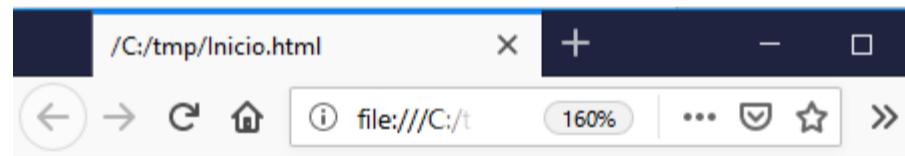


Ilustración 49: Mensaje de error por la sintaxis errónea

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.

098.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    let lenguajes = []; //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>
</body>
</html>
```

099.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    let lenguajes = []; //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 (let cont = 0; cont < lenguajes.length; cont++) {
        document.write("<br>" + lenguajes[cont]);
    }
</script>
</body>
</html>
```

100.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    let 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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Otra forma de declarar arreglos y mostrarlos
    let pagar = [1120, 6040, 1570, 5205, 3147, 7361, 166, 1002, 1572, 6567];
    document.write(pagar);
</script>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
let lenguajes = []; //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>
</body>
</html>
```

Operaciones con arreglos

103.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    let valores = [];
    valores.push(15000);
    valores.push(18000);
    valores.push(34000);
    valores.push(17000);
    valores.push(8000);

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

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

104.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Operación con vectores: promedio
    let valores = [];
    valores.push(15000);
    valores.push(18000);
    valores.push(34000);
    valores.push(17000);
    valores.push(8000);

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

    //Calcula el promedio
    let promedio = acumula / valores.length;
    document.write("Promedio es: " + promedio);
</script>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Operación con vectores: Buscar el mayor valor
    const valores = [];
    valores.push(15000);
    valores.push(18000);
    valores.push(34000);
    valores.push(17000);
    valores.push(8000);

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

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ordena arreglos con valores alfanuméricos
    let lenguajes = [];
    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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ordenar de menor a mayor el arreglo con valores numéricos
    let valores = [];
    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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Ordena en forma inversa un arreglo con valores alfanuméricos
    let lenguajes = [];
    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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Dado un arreglo extraer los valores únicos
    let 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];
    let extrae = [];

    for (let 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 (let cont = 0; cont < arreglo.length; cont++)
            if (num === arreglo[cont])
                return true;
        return false;
    }
</script>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    // ¿Función genérica para hallar menor y mayor valor?
    let pagar = [1120, 6040, 1570, 5205, 3147, 7361, 166, 1002, 1572, 6567];

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

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

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

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

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

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

Implementación de algoritmos de ordenación

Algoritmo de burbuja

112.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    let 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 (let i = 0; i < arreglo.length - 1; i++)
            for (let j = 0; j < arreglo.length - i - 1; j++)
                if (arreglo[j] > arreglo[j + 1]) {
                    let aux = arreglo[j];
                    arreglo[j] = arreglo[j + 1];
                    arreglo[j + 1] = aux;
                }
    }
</script>
</body>
</html>
```

113.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Arreglo de números enteros
    let 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 (let i = 0; i < arreglo.length - 1; i++)
            for (let j = 0; j < arreglo.length - i - 1; j++)
                if (arreglo[j] > arreglo[j + 1]) {
                    let aux = arreglo[j];
                    arreglo[j] = arreglo[j + 1];
                    arreglo[j + 1] = aux;
                }
    }
</script>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    let 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) {
        let intercambio;
        do {
            intercambio = false;
            for (let i = 0; i < arreglo.length - 1; i++) {
                if (arreglo[i] > arreglo[i + 1]) {
                    let temporal = arreglo[i];
                    arreglo[i] = arreglo[i + 1];
                    arreglo[i + 1] = temporal;
                    intercambio = true;
                }
            }
        } while (intercambio);
    }
</script>
</body>
</html>
```

Algoritmo de ordenación por selección

115.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    let 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 (let i = 0; i < arreglo.length - 1; i++) {
            let tmp = i;
            for (let j = i + 1; j < arreglo.length; j++)
                if (arreglo[j] < arreglo[tmp])
                    tmp = j;

            let temporal = arreglo[tmp];
            arreglo[tmp] = arreglo[i];
            arreglo[i] = temporal;
        }
    }
</script>
</body>
</html>
```

116.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    let 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 (let i = 0; i < arreglo.length - 1; i++) {
            let tmp = i;
            for (let j = i + 1; j < arreglo.length; j++)
                if (arreglo[j] < arreglo[tmp])
                    tmp = j;

            let temporal = arreglo[tmp];
            arreglo[tmp] = arreglo[i];
            arreglo[i] = temporal;
        }
    }
</script>
</body>
</html>
```

Algoritmo de ordenación por inserción

117.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
let 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 (let 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>
</body>
</html>
```

118.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
let 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 (let 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>
</body>
</html>
```

Algoritmo de ordenación QuickSort

119.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
let letras = ['r', 'v', 'e', 'g', 't', 'u', 'd', 'f', 'f', 'y', 'v'];
document.write("Original: " + letras + "<br>");
let ordenado = quicksort(letras);
document.write("Ordenado: " + ordenado + "<br>");

//Algoritmo de ordenación QuickSort
function quicksort(arreglo) {
    if (arreglo.length === 0) return [];
    let izquierdo = [], derecho = [], pivote = arreglo[0];
    for (let 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>
</body>
</html>
```

120.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
let datos = [-363, -1793, -2601, -2541, 6376, 6053, 503, 1601, -2994, 648];
document.write("Original: " + datos + "<br>");
let ordenado = quicksort(datos);
document.write("Ordenado: " + ordenado + "<br>");

function quicksort(arreglo) {
    if (arreglo.length === 0) return [];
    let izquierdo = [], derecho = [], pivote = arreglo[0];
    for (let 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>
</body>
</html>
```

Algoritmo de ordenación MergeSort

121.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
let letras = ['r', 'v', 'e', 'g', 't', 'u', 'd', 'f', 'f', 'y', 'v'];
document.write("Original: " + letras + "<br>");
let ordenado = mergesort(letras);
document.write("Ordenado: " + ordenado + "<br>");

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

function merge(izquierdo, derecho) {
    let ordenado = [];
    while (izquierdo && izquierdo.length > 0 && derecho && derecho.length > 0) {
        let 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>
</body>
</html>
```

122.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
let datos = [-363, -1793, -2601, -2541, 6376, 6053, 503, 1601, -2994, 648];
document.write("Original: " + datos + "<br>");
let ordenado = mergesort(datos);
document.write("Ordenado: " + ordenado + "<br>");

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

function merge(izquierdo, derecho) {
    let ordenado = [];
    while (izquierdo && izquierdo.length > 0 && derecho && derecho.length > 0) {
        let 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>
</body>
</html>
```

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

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

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Concatenar cadenas
    let cadena1 = "Dragón";
    let cadena2 = " de ";
    let cadena3 = "Komodo";
    let cadena4 = cadena1 + cadena2 + cadena3;
    document.write(cadena4);
</script>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Concatenar cadenas y valores numéricos
    let cadena1 = 3;
    let cadena2 = "abcde";
    let cadena3 = 5;
    let cadena4 = cadena1 + cadena2 + cadena3 + "efg";
    document.write(cadena4);
</script>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Extraer una letra de una cadena. Las cadenas se comportan como arreglos unidimensionales, el primer
    elemento está en la posición 0 */
    let cadena = "Esta es una prueba";
    let letra = cadena[0]; //Trae la primera letra
    document.write(letra);
</script>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Longitud de una cadena (incluye los espacios)
    let cadena = "    Esta    es    una    prueba    ";
    let longitud = cadena.length;
    document.write(longitud);
</script>
</body>
</html>
```

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Comparación de cadenas
    let cadena1 = "FraseA";
    let cadena2 = "FraseA";
    if (cadena1 === cadena2) //Comparación estricta al usar ===
        document.write("iguales");
    else
        document.write("diferentes");
</script>
</body>
</html>
```

Ejemplos de programas usando cadenas

Invertir Cadena

132.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Invierte la cadena
    let cadena = "Rafael Alberto Moreno Parra";
    let nueva = "";
    for (let cont = 0; cont < cadena.length; cont++) nueva = cadena[cont] + nueva;

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

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

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Quitar vocales (minúsculas, mayúsculas, tildadas)
    let cadena = prompt("Escriba un texto");
    let nueva = "";
    for (let 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 carácter es una vocal
        let vocales = "aeiouAEIOUÁÉÍÓÚáéíóúäëöü";
        for (let letra = 0; letra < vocales.length; letra++)
            if (caracter === vocales[letra])
                return true;

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Retirar caracteres no permitidos de una cadena
    let cadena = "Esto es una <b>prueba</b>";
    let nueva = "";
    for (let 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 carácter está permitido
    function EsPermitido(caracter) {
        let permite = "ÁÉÍÓÚáéíóúäëöü";
        permite += "abcdefghijklmnñopqrstuvwxyz";
        permite += "ABCDEFGHIJKLMNÑOPQRSTUVWXYZ";
        permite += " 1234567890";
        for (let letra = 0; letra < permite.length; letra++)
            if (caracter === permite[letra])
                return true;
        return false;
    }
</script>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
//Cifrado sencillo
let cadena = prompt("Escriba un texto");
let cifrada = Cifrar(cadena);
document.write("Original: " + cadena + "<br>");
document.write("Cifrado: " + cifrada + "<br>");

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

//Cifra la cadena moviendo un carácter hacia adelante
function Cifrar(cadena) {
    let cifrada = "";
    for (let cont = 0; cont < cadena.length; cont++) {
        let 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) {
    let descifrada = "";
    for (let cont = 0; cont < cadena.length; cont++) {
        let 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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Cifrado sencillo con clave de cifrado
    let cadena = prompt("Escriba texto");
    let clave = prompt("Clave de cifrado");

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

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

    function Cifrar(cadena, clave) {
        let cifrada = "";
        let clavecont = 0;
        for (let cont = 0; cont < cadena.length; cont++) {
            //Cada letra de la clave genera un desplazamiento
            let 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) {
        let descifrada = "";
        let clavecont = 0;
        for (let cont = 0; cont < cadena.length; cont++) {
            //Cada letra de la clave genera un desplazamiento
            let 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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Suma dos números enteros positivos almacenados en cadenas
    let numeroA = "395763487398655335"; //Primer número
    let numeroB = "8888888888888888"; //Segundo número
    let resultado = ""; //Guarda el resultado
    let ultimaPosA = numeroA.length - 1; //Posición del último dígito del primer número
    let ultimaPosB = numeroB.length - 1;
    let llevar = false; //Se activa si la suma de dígitos es 10 o más
    while (ultimaPosA >= 0 || ultimaPosB >= 0) { //Suma del dígito menos significativo al más
        significativo
            let 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++;
            llevar = false; //La bandera de llelet se apaga
            if (suma >= 10) {
                llevar = true;
                suma -= 10;
            }
            resultado = suma + resultado; //Agrega al inicio de la cadena
            ultimaPosA--;
            ultimaPosB--;
    }
    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>
</body>
</html>
```

Arreglos bidimensionales

Con números

139.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Crea el arreglo bidimensional
    let bidiarreglo = new Array(2); //Crea dos filas

    //Crea cuatro columnas por fila
    for (let 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 (let fila = 0; fila < bidiarreglo.length; fila++) {
        document.write("<br>");
        for (let columna = 0; columna < bidiarreglo[fila].length; columna++)
            document.write(bidiarreglo[fila][columna] + " , ");
    }
</script>
</body>
</html>
```

Con caracteres

140.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Crea el arreglo bidimensional
    let tabla = new Array(8); //Crea las filas

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

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

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

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

Función genérica

141.html

```
<!DOCTYPE HTML><html lang="es"><head><title>Capacitándose en JavaScript</title></head>
<body>
<script>
    //Llama a función que genera arreglo bidimensional
    let tabla = GenerarArregloBidimensional(8);

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

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

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

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

Funciones genéricas

142.html

```
<!DOCTYPE HTML><html lang="es"><head><title>Capacitándose en JavaScript</title></head>
<body>
<script>
    let 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) {
        let arreglo = [];
        for (let fila = 0; fila < filas; fila++) {
            arreglo[fila] = [];
            for (let columna = 0; columna < columnas; columna++)
                arreglo[fila][columna] = valor;
        }
        return arreglo;
    }

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

Ejemplos de programas usando arreglos bidimensionales

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

143.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Crear un tablero de ajedrez de 8*8, poner en una posición al
       azar una Torre y mostrar su ataque */

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

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

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

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

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Crear un tablero de ajedrez de 8*8, poner en una posición al
       azar el Rey y mostrar su ataque */
    let tabla = GeneraArregloBidimensional(8, 8, '.');
    PoneReyAzar(tabla);
    ImprimeArregloBidimensional(tabla);

    //Pone un Rey al azar y muestra su ataque
    function PoneReyAzar(arreglo) {
        let filaAzar = Math.floor(Math.random() * arreglo.length);
        let 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) {
        let arreglo = [];
        for (let fila = 0; fila < filas; fila++) {
            arreglo[fila] = [];
            for (let columna = 0; columna < columnas; columna++)
                arreglo[fila][columna] = valor;
        }
        return arreglo;
    }

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

```

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Crear un tablero de ajedrez de 8*8, poner en una posición al
       azar un Alfil y mostrar su ataque */
    let tabla = GeneraArregloBidimensional(8, 8, '.');
    PoneAlfilAzar(tabla);
    ImprimeArregloBidimensional(tabla);

    function PoneAlfilAzar(arreglo) {
        let filaAzar = Math.floor(Math.random() * arreglo.length);
        let coluAzar = Math.floor(Math.random() * arreglo[0].length);
        let tmpFil = filaAzar;
        let 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) {
        let arreglo = [];
        for (let fila = 0; fila < filas; fila++) {
            arreglo[fila] = [];
            for (let columna = 0; columna < columnas; columna++)
                arreglo[fila][columna] = valor;
        }
        return arreglo;
    }

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

```

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Crear un tablero de ajedrez de 8*8, poner en una posición al
       azar la Reina y mostrar su ataque */
    let tabla = GeneraArregloBidimensional(8, 8, '.');
    PoneReinaAzar(tabla);
    ImprimeArregloBidimensional(tabla);

    function PoneReinaAzar(arreglo) {
        let filaAzar = Math.floor(Math.random() * arreglo.length);
        let coluAzar = Math.floor(Math.random() * arreglo[0].length);
        let tmpFil = filaAzar;
        let 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) {
        let arreglo = [];
        for (let fila = 0; fila < filas; fila++) {
            arreglo[fila] = [];
            for (let columna = 0; columna < columnas; columna++)
                arreglo[fila][columna] = valor;
        }
        return arreglo;
    }

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

```

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.

147.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Hay N ciudades a recorrer (0 a N-1). Sólo se puede visitar una ciudad por vez.
       En la tabla aparece cuanto cuesta ir de una ciudad (origen) a otra ciudad (destino).
       ¿Qué ruta tomar para visitar todas las ciudades con el mínimo costo? */

    let ciudad = 20; //Número de ciudades
    let minValue = 15; //Valor mínimo que tendrá ir de una ciudad a otra
    let maxValue = 85; //Valor máximo que tendrá ir de una ciudad a otra

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

    //Genera valores de viaje al azar
    let valorviajes = inicializavalorviajes(ciudad, minValue, maxValue);

    //Imprime los valores
    imprime(valorviajes);

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

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

    //Usando el método Monte Carlo se buscarán otras rutas con menor costo
    for (let pruebas = 1; pruebas <= 500000; pruebas++) {
        dest1 = Math.floor(Math.random() * ruta.length);
        do {
            dest2 = Math.floor(Math.random() * ruta.length);
        } while (dest2 === dest1)
        ModificaRuta(ruta, dest1, dest2)
        let 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) {
        let tmp = ruta[dest1];
        ruta[dest1] = ruta[dest2];
        ruta[dest2] = tmp;
    }

    //Inicia el arreglo bidimensional de rutas
    function iniciaRuta(limite) {
        const ruta = [];
        for (let cont = 0; cont < limite; cont++) ruta.push(cont);
        return ruta;
    }

    //Deduce el costo de la ruta de viaje
    function DeduceCosto(ruta, costos) {
        let acum = 0;
        for (let 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, minValue, maxValue) {
        let tablero = [];
        for (let fila = 0; fila < ciudad; fila++) {

```

```

        tablero[fila] = [];
        for (let columna = 0; columna < ciudad; columna++)
            tablero[fila][columna] = Math.floor(Math.random() * (maxValor - minValor) + minValor);
    }
    for (let cont = 0; cont < ciudad; cont++) tablero[cont][cont] = 0;
    return tablero;
}

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

```

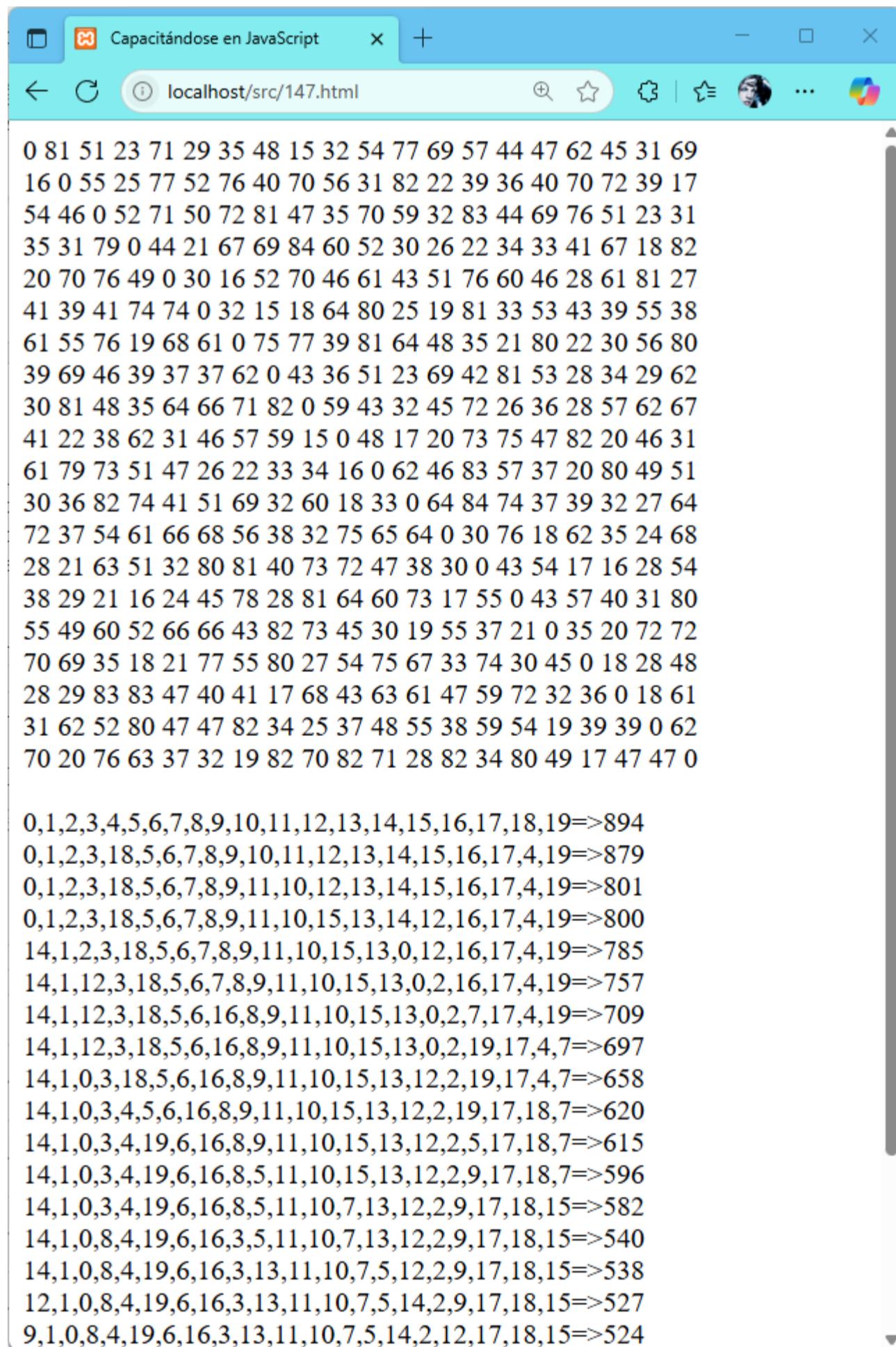


Ilustración 50: Ruta del menor costo. Genera nuevas rutas más económicas

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>

let 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]
];

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

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

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

    let numValido = true;
    do { /* Busca un número al azar para colocar en alguna celda */
        let posX = Math.floor(Math.random() * 9); /* Una posición X de 0 a 8 */
        let posY = Math.floor(Math.random() * 9); /* Una posición Y de 0 a 8 */
        let 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 (let 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 (let cuadroX = 0; cuadroX <= 6; cuadroX += 3)
        for (let cuadroY = 0; cuadroY <= 6; cuadroY += 3) {
            let numRepite = 0;
            for (let valor = 1; valor <= 9; valor++) {
                numRepite = 0;
                for (let posX = 0; posX < 3; posX++)
                    for (let 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 (let posX = 0; posX < 3; posX++)
                        for (let 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 (let posX = 0; posX < 9; posX++)
        for (let posY = 0; posY < 9; posY++)
            if (solucion[posX][posY] === 0) finalizo = false;
}

ciclos++; /* Cada ciertos ciclos para destrabar, borra la tercera parte de lo completado */

```

```

if (ciclos % DESTRUYE === 0)
    for (let posX = 0; posX < 9; posX++)
        for (let 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 (let posX = 0; posX < 9; posX++) {
    document.write("<p style='font-family:courier new;'>");
    for (let posY = 0; posY < 9; posY++)
        document.write(solucion[posX][posY] + " ");
    document.write("</p>");
}
</script>
</body>
</html>

```



Ilustración 51: Sudoku resuelto

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Hacer un algoritmo que ponga los 5 barcos
       al azar en el tablero del juego de batalla naval */

    //Crear el tablero del juego
    let 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) {
        let tablero = [];
        for (let fila = 0; fila < filas; fila++) {
            tablero[fila] = [];
            for (let columna = 0; columna < columnas; columna++)
                tablero[fila][columna] = '.';
        }
        return tablero;
    }

    function imprime(tablero) { //Imprime el tablero
        for (let fila = 0; fila < tablero.length; fila++) {
            document.write("<p style='font-family:courier new;'>");
            for (let 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) {
        let 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);
            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 (let cont = 0; cont < huecos; cont++) {
                        if (tablero[fil + cont][col] !== '.')
                            posValida = false;
                    }
                else
                    for (let cont = 0; cont < huecos; cont++) {
                        if (tablero[fil][col + cont] !== '.')
                            posValida = false;
                    }
            }
        } while (posValida === false);

        if (orienta < 0.5)
            for (let cont = 0; cont < huecos; cont++)
```

```
    tablero[fil + cont][col] = simbolo;
} else
    for (let cont = 0; cont < huecos; cont++)
        tablero[fil][col + cont] = simbolo;
}
</script>
</body>
</html>
```

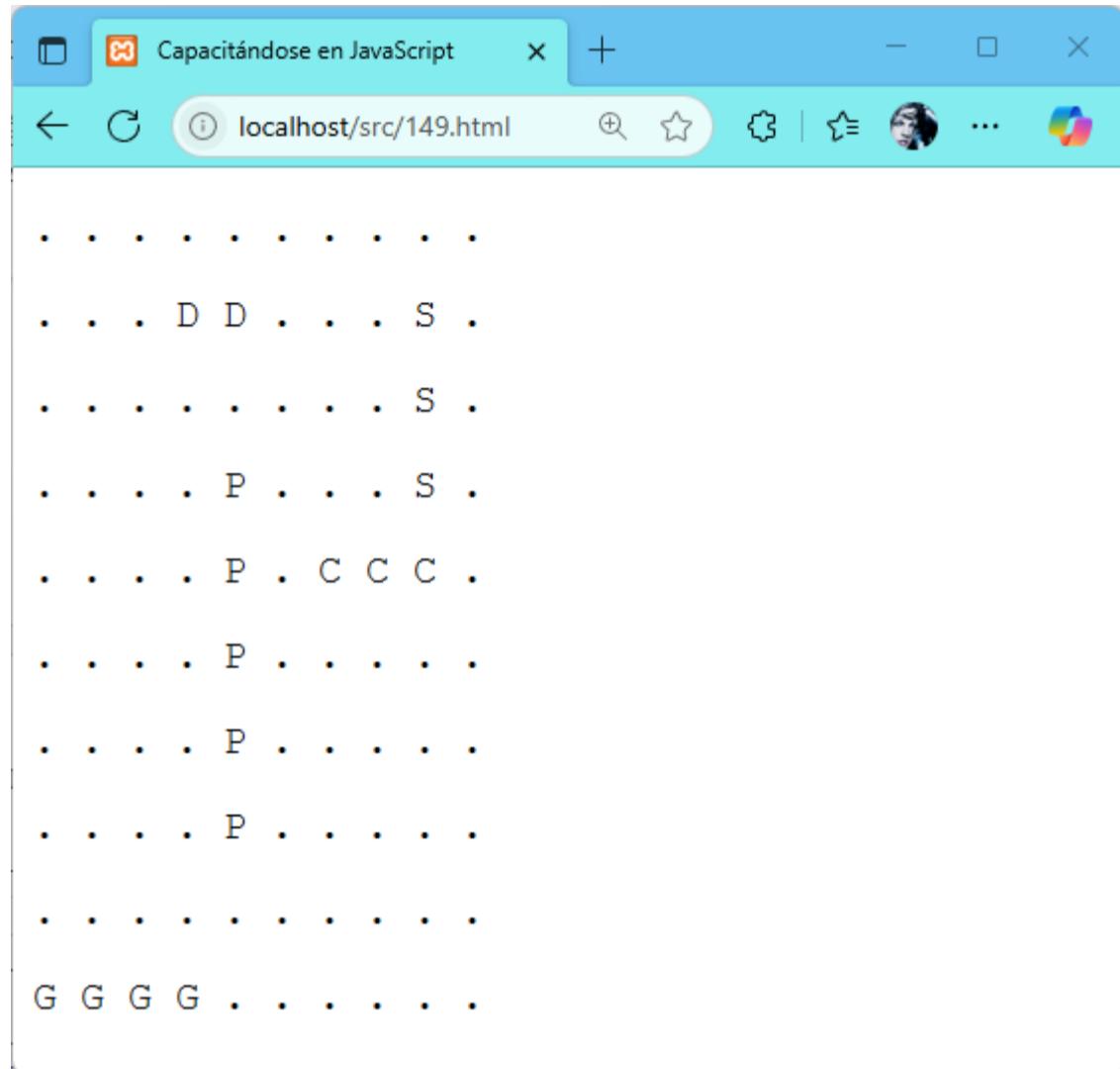


Ilustración 52: Barcos ubicados en el juego de mesa Batalla Naval

Funciones que reciben distinto número de parámetros

150.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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>
</body>
</html>
```

151.html

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

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Envia una función como parámetro a otra función

    //Se envía una función que retorna el resultado de a - b
    let 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) {
        let valorA = 15;
        let valorB = 10;
        //Dependiendo de la función recibida como parámetro, ejecuta la acción
        return unafuncion(valorA, valorB);
    }
</script>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Envia una función como parámetro a otra función
    let 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) {
        return unafuncion(valorA, valorB);
    }
</script>
</body>
</html>
```

Operaciones de bit

Convertir un entero a su representación binaria

154.html

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    let valA = 64;
    let 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>
</body>
</html>
```

Programación orientada a objetos

Se sigue la especificación ECMAScript 2025

Definiendo clases, constructores y atributos

161.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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
    let prueba = new Gato("Sally", "F", "10 de junio de 2010", "Criollo");

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

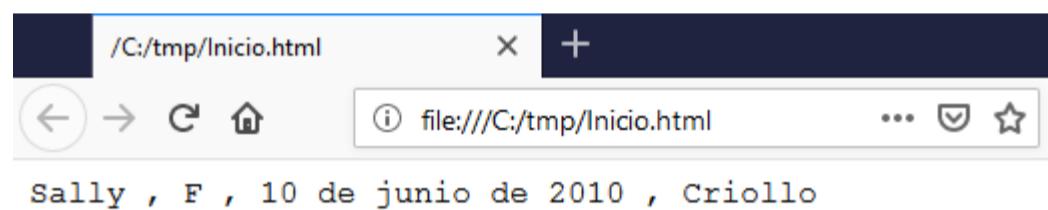


Ilustración 53: Valores de atributos

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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
    let probar = new Geometria();

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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
    let probar = new Geometria();
    probar.ValorLado(11);
    document.write("Área del Cuadrado es: " + probar.AreaCuadrado());
</script>
</body>
</html>
```

Atributos y métodos privados

Se hace uso del carácter # para definir los atributos y métodos privados

164.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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
    let probar = new UnaClase();
    probar.#UnMetodo();
    probar.#OtroMetodo();
</script>
</body>
</html>
```

Si intenta acceder a un método privado desde la instancia, se genera un mensaje de error

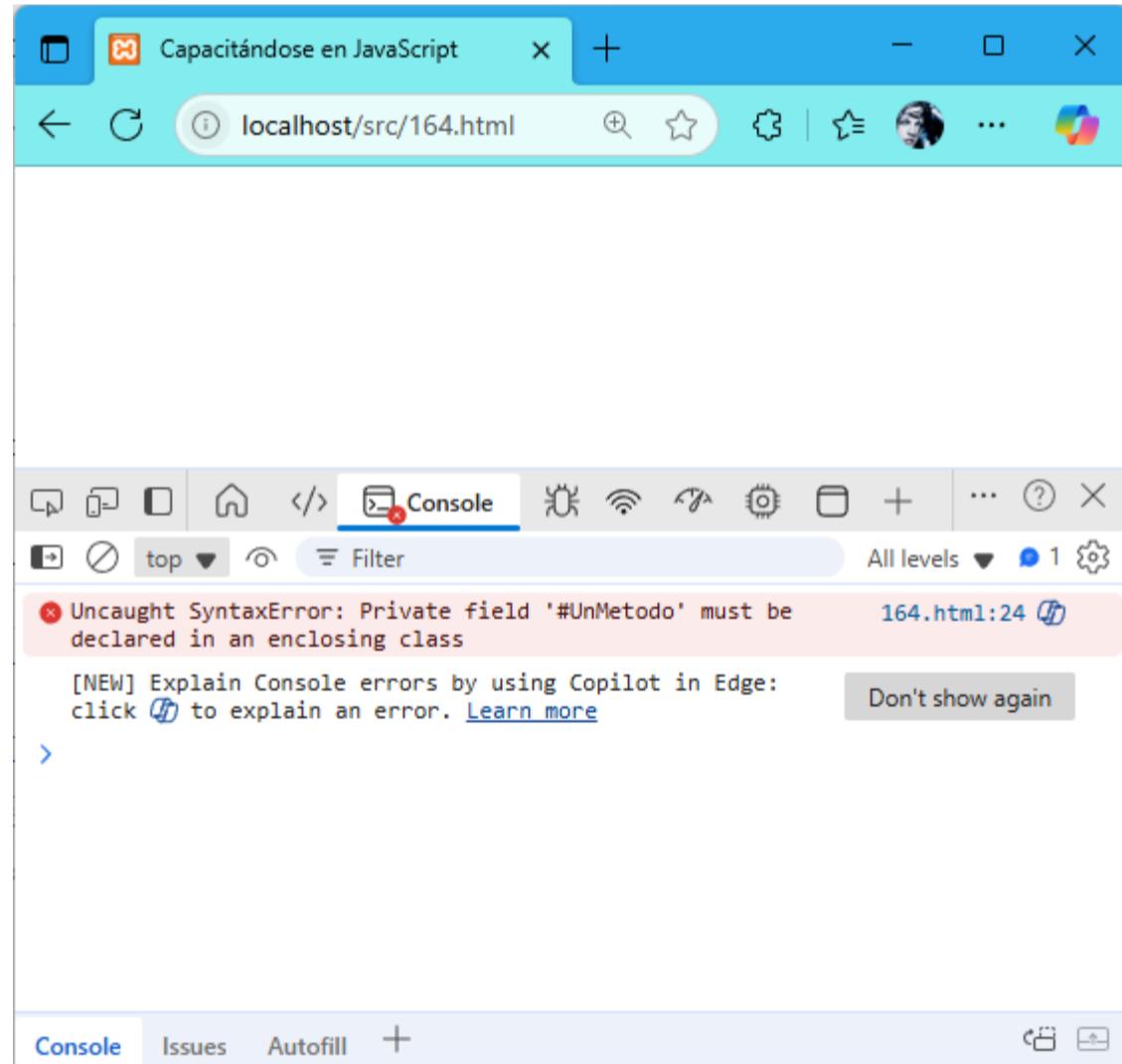


Ilustración 54: Métodos privados

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    class MiClase {
        #texto; //Declara el atributo privado

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

Podemos probar que los métodos get y set son llamados

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    class MiClase {
        #texto; //Declara el atributo privado

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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>
</body>
</html>
```

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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>
</body>
</html>

```

iOJO! 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

Sobrecarga de métodos

No hay sobrecarga de métodos como tal en JavaScript. El siguiente script pareciera que se implementa sobrecarga, pero NO funciona. Lo peor, es que no se genera mensaje de error en el navegador.

170.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* La sobrecarga de métodos 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>
</body>
</html>
```

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

171.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
  <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
  let 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>
</body>
</html>
```

172.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
  <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
  class MiClase {
  }

  prueba = new MiClase();

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

Árbol binario

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

173.html

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

    let NodoRaiz = new Nodo(1, null, null); //Nodo raíz
    let NodoIzq = new Nodo(2, null, null); //Rama izquierda
    let 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>
</body>
</html>
```

Recorrido en pre-orden

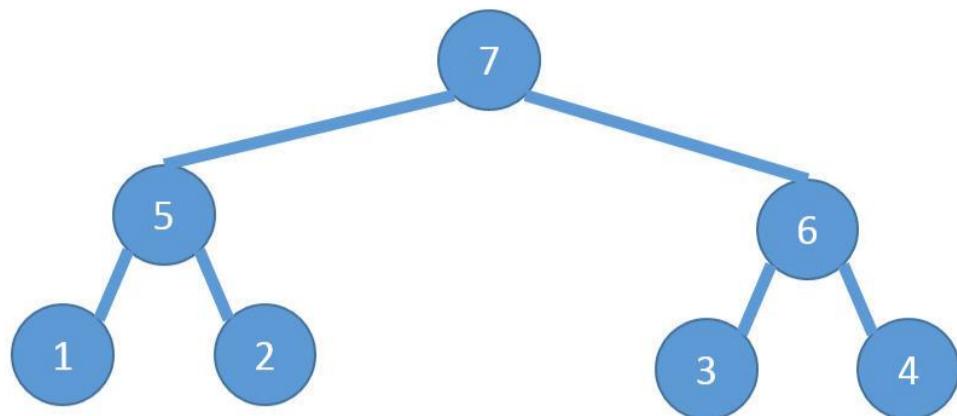


Ilustración 55: Árbol binario

174.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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
    let Nodo1 = new Nodo(1, null, null);
    let Nodo2 = new Nodo(2, null, null);
    let Nodo3 = new Nodo(3, null, null);
    let Nodo4 = new Nodo(4, null, null);
    let Nodo5 = new Nodo(5, Nodo1, Nodo2);
    let Nodo6 = new Nodo(6, Nodo3, Nodo4);
    let 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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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
    let Nodo1 = new Nodo(1, null, null);
    let Nodo2 = new Nodo(2, null, null);
    let Nodo3 = new Nodo(3, null, null);
    let Nodo4 = new Nodo(4, null, null);
    let Nodo5 = new Nodo(5, Nodo1, Nodo2);
    let Nodo6 = new Nodo(6, Nodo3, Nodo4);
    let 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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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
    let Nodo1 = new Nodo(1, null, null);
    let Nodo2 = new Nodo(2, null, null);
    let Nodo3 = new Nodo(3, null, null);
    let Nodo4 = new Nodo(4, null, null);
    let Nodo5 = new Nodo(5, Nodo1, Nodo2);
    let Nodo6 = new Nodo(6, Nodo3, Nodo4);
    let 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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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
    let arreglo = [15, 21, 17, 32, 48, 13, 29, 44, 19, 16, 3, -1, 9, 0];

    //Genera el árbol binario
    let NodoRaiz = new Nodo(arreglo[0], null, null);
    for (let cont = 1; cont < arreglo.length; cont++) {
        let 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>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    class Nodo {
        constructor(valorA, valorB) {
            this.valA = valorA;
            this.valB = valorB;
        }
    }

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    class Nodo {
        constructor(valorA, valorB) {
            this.valA = valorA;
            this.valB = valorB;
        }
    }

    //Trabaja como una pila
    let lista = [];
    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 (let cont = 0; cont < lista.length; cont++)
        document.write(lista[cont].valA + ", " + lista[cont].valB + "<br>");
</script>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    class Nodo {
        constructor(valorA, valorB) {
            this.valA = valorA;
            this.valB = valorB;
        }
    }

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    class Nodo {
        constructor(valorA, valorB) {
            this.valA = valorA;
            this.valB = valorB;
        }
    }

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

Instrucciones que ejecutan controlando el tiempo

182.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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>
</body>
</html>
```

183.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Cada 100 milisegundos se llamará a la función Imprime()
    setInterval(Imprime, 100);

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

184.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    let contador = 1;

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

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

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

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</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 */
    let 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 (let divide = 3; divide <= Math.sqrt(num); divide += 2)
            if (num % divide === 0) return false;
        return true;
    }
</script>
</body>
</html>

```

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</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. */
    let numero = 10000;
    let ejecuta = setInterval(Imprime, 100);
    let 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 (let divide = 3; divide <= Math.sqrt(num); divide += 2)
            if (num % divide === 0) return false;
        return true;
    }
</script>
</body>
</html>

```

```
<!DOCTYPE HTML><html lang="es"><head><title>Capacitándose en JavaScript</title></head>
<body>
<script>
    //Primer ejemplo de uso de JSON
    let 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>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Segundo ejemplo de uso de JSON
    let dato = {
        "mensaje": function () {
            document.write("Hola Mundo");
        }
    }
    dato.mensaje();

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Tercer ejemplo de uso de JSON
    let dato = {
        "nombre": "Rafael", "mensaje": function () {
            document.write(dato.nombre);
        }
    }
    dato.mensaje();

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Cuarto ejemplo de uso de JSON
    let 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>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Quinto ejemplo de uso de JSON
    let animal =
    {
        "Aves": [
            {"nombre": "Cuervo", "esperanzavida": 70},
            {"nombre": "Loro Gris", "esperanzavida": 80},
            {"nombre": "Zopilote de Turqui", "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>
```

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Sexto ejemplo de uso de JSON
    let 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 (let 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>

```

```

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

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

        //va de ítem en ítem de cada grupo de temperaturas
        for (let 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>

```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Octavo ejemplo de uso de JSON
    let 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>
```

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"

195.html

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

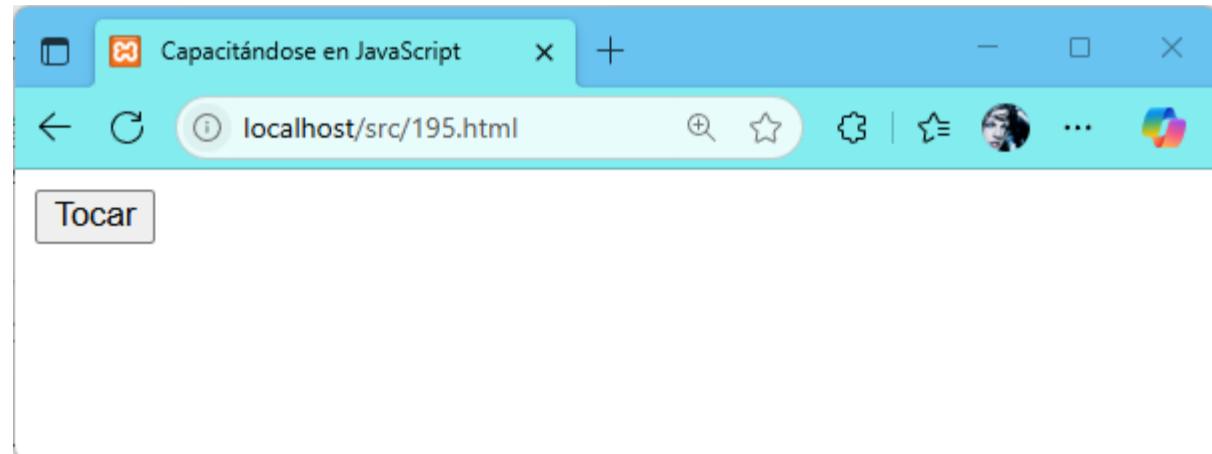


Ilustración 56: Botón

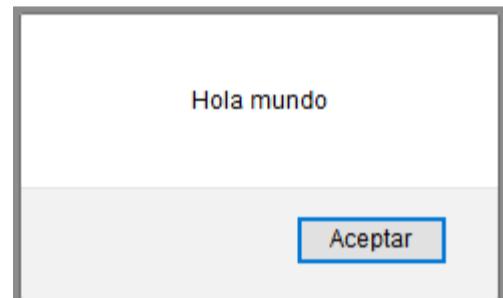


Ilustración 57: Ventana emergente generada por JavaScript

El siguiente código hace lo mismo haciendo uso de funciones

196.html

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

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

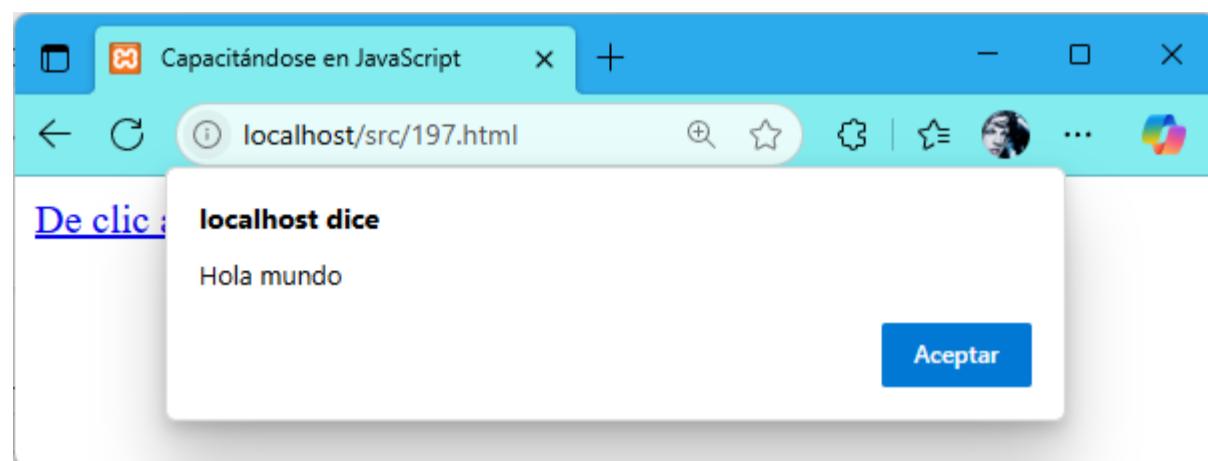


Ilustración 58: Ventana emergente al dar clic en un hipervínculo

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

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

198.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <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>

199.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <style>
        div {
            width: 230px;
            height: 230px;
            margin: 20px;
            background-color: lightgreen;
        }
    </style>
    <title>Capacitándose en JavaScript</title>
    <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>
```

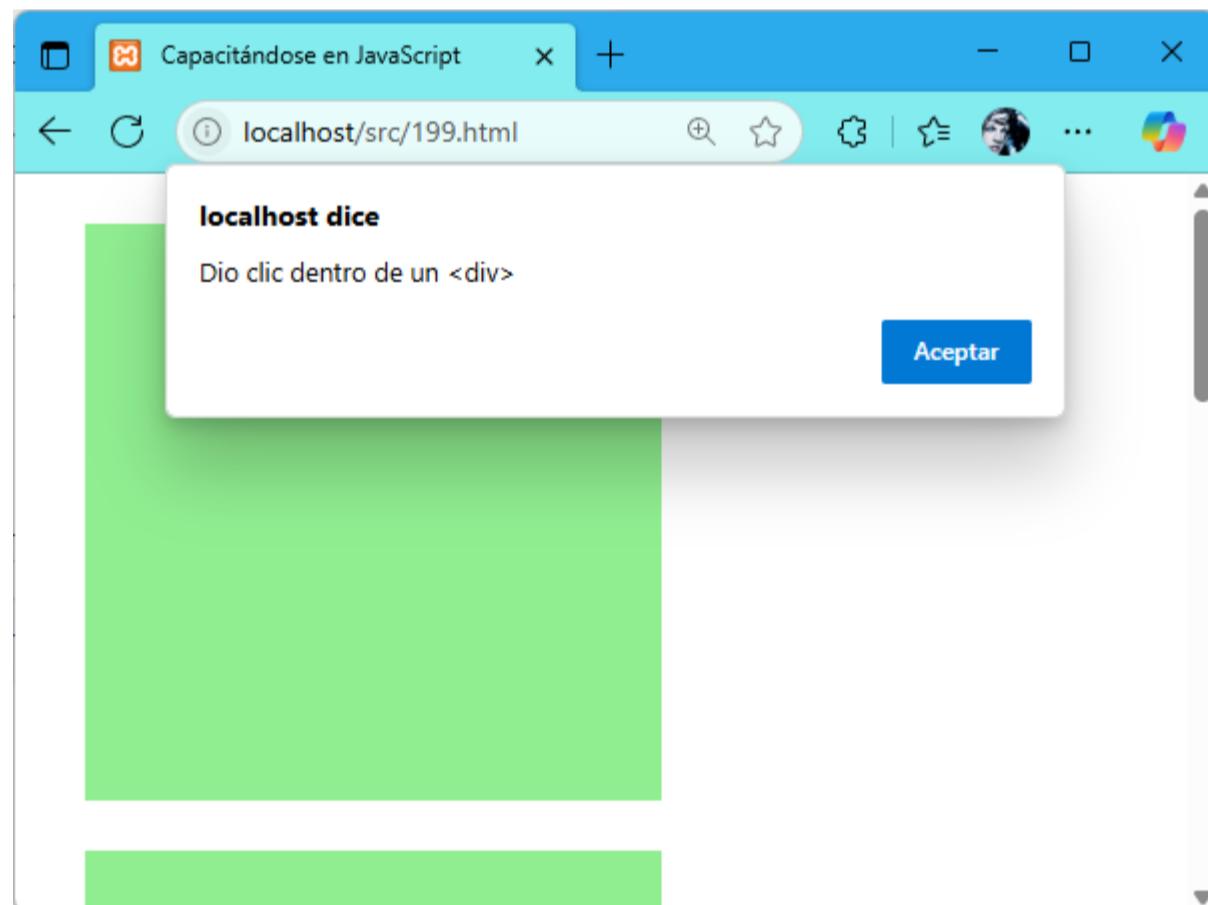


Ilustración 59: Ventana emergente al dar clic en un <div>

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <style>
        div {
            width: 70px;
            height: 70px;
            margin: 20px;
            background-color: lightgreen;
        }
    </style>
</head>
<body>
<script>
    function divclic(caja) {
        alert("<div> tocado es: " + caja.id);
    }
</script>
<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>
```

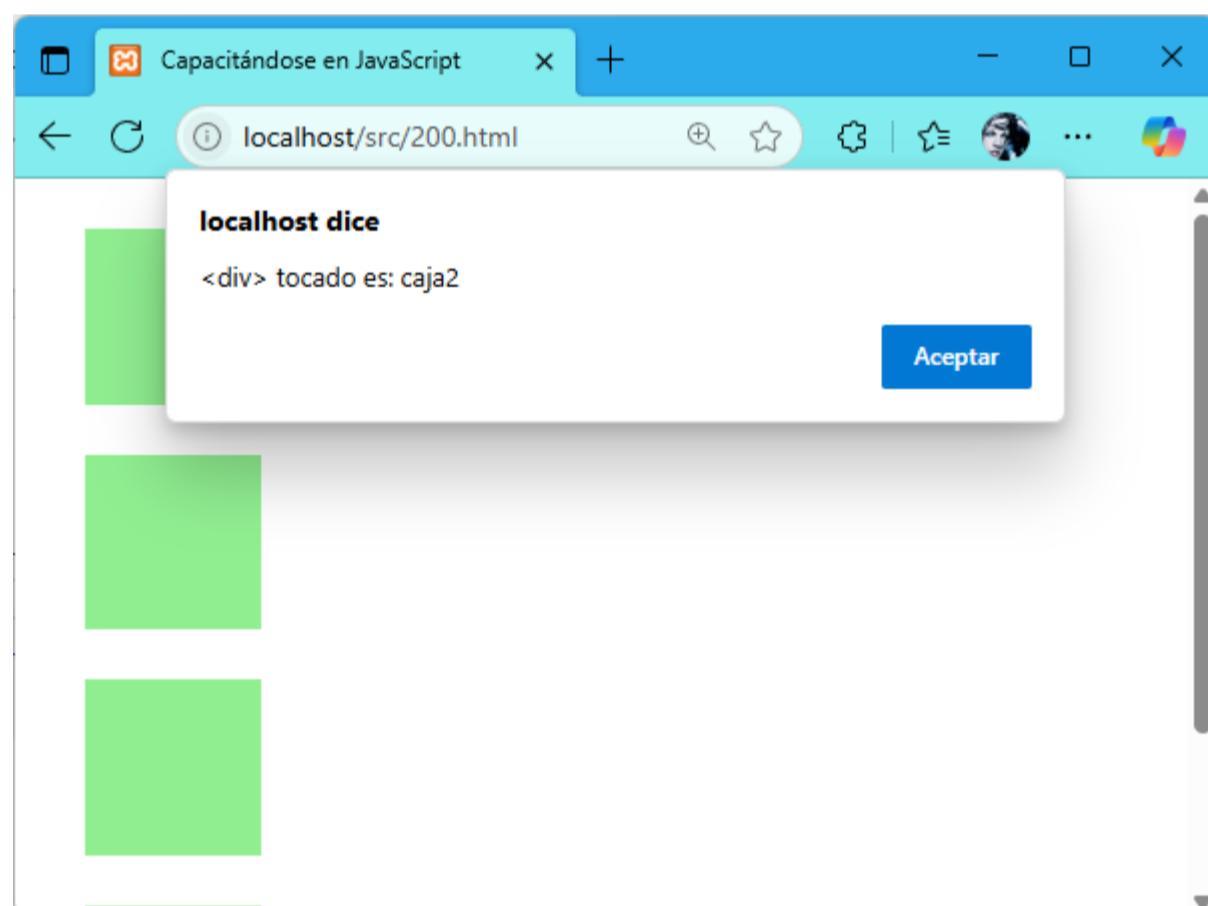


Ilustración 60: Ventana emergente al dar clic en un <div>

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

201.html

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

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

202.html

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

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

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

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

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

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>

207.html

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body onresize="cambialongitud()">
<script>
    function cambialongitud() {
        alert("¡Crecer o decrecer!");
    }
</script>
<p>Cambio el tamaño de la ventana del navegador</p>
</body>
</html>
```

Captura el evento de dar “submit” en un formulario

209.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    function formularioenvia() {
        alert("activo envío");
    }
</script>
<form id="formulario" method="post" onsubmit="formularioenvia()">
    <input type="submit" id="boton" value="envía"/>
</form>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    function formularioenvia() {
        alert("activo envío");
    }

    function entrafocotexto() {
        alert("está en la caja de texto");
        document.getElementById("otro").focus();
    }
</script>
<form id="formulario" method="post" onsubmit="formularioenvia()">
    <input type="text" id="texto" onfocus="entrafocotexto()"/><br>
    <input type="text" id="otro" /><br>
    <input type="submit" id="boton" value="envía"/>
</form>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    function formularioenvia() {
        alert("activo envío");
    }

    function salefocotexto() {
        alert("salió de la caja de texto");
    }
</script>
<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>
```

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

212.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    function formularioenvia() {
        alert("activo envío");
    }

    function presionatecla() {
        alert("Ha presionado una tecla");
    }
</script>
<form id="formulario" method="post" onsubmit="formularioenvia()">
    <input type="text" id="texto" onkeydown="presionatecla()" />
    <input type="submit" id="boton" value="envía"/>
</form>
</body>
</html>
```

Un efecto similar sucede con "onkeypress"

213.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    function formularioenvia() {
        alert("activo envío");
    }

    function presionatecla() {
        alert("Ha presionado una tecla");
    }
</script>
<form id="formulario" method="post" onsubmit="formularioenvia()">
    <input type="text" id="texto" onkeypress="presionatecla()" />
    <input type="submit" id="boton" value="envía"/>
</form>
</body>
</html>
```

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

214.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    function formularioenvia() {
        alert("activo envío");
    }

    function abajo() {
        alert("abajo");
    }

    function presiona() {
        alert("presiona");
    }
</script>
<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>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    function formularioenvia() {
        alert("activo envío");
    }

    function sueltatecla() {
        alert("Ha soltado una tecla");
    }
</script>
<form id="formulario" method="post" onsubmit="formularioenvia()">
    <input type="text" id="texto" onkeyup="sueltecla()"/>
    <input type="submit" id="boton" value="envía"/>
</form>
</body>
</html>
```

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

216.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    function formularioenvia() {
        alert("activo envío");
    }

    function cambiatexto() {
        alert("Ha cambiado el texto");
    }
</script>
<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>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    function formularioenvia() {
        alert("envío");
    }

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    function navegar() {
        return confirm("¿Irá a esa página?");
    }
</script>
<a href="http://darwin.50webs.com" onclick="return navegar()">Enlace</a>
</body>
</html>
```

DOM (Document Object Model)

Cambiar atributos de un objeto

219.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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
        let enlace = document.getElementById("ejemploenlace");

        //Cambia hacia donde enlaza
        enlace.href = "http://darwin.50webs.com";
    }
</script>
<a id="ejemploenlace">Hipervínculo</a>
<input type="button" onclick="enlazarmipagina()" value="Mi página"/>
</body>
</html>
```

Cambiar el color de fondo de un <div>

220.html

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <style>
        #caja {
            width: 200px;
            height: 200px;
            background-color: lightgreen;
        }
    </style>
</head>
<body>
<script>
    //Oculta el <div>
    function apagar() {
        let caja = document.getElementById("caja");
        caja.style.visibility = "hidden";
    }

    //Muestra el <div>
    function encender() {
        let caja = document.getElementById("caja");
        caja.style.visibility = "visible";
    }
</script>
<div id="caja"></div>
<input type="button" value="Apagar" onclick="apagar()" />
<input type="button" value="Encender" onclick="encender()" />
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <style>
        #caja {
            width: 200px;
            height: 200px;
            background-color: lightgreen;
        }
    </style>
</head>
<body>
<script>
    //Borra contenido del <div>
    function borrar() {
        let caja = document.getElementById("caja");
        caja.innerHTML = "";
    }

    //Pone contenido en el <div>
    function texto() {
        let caja = document.getElementById("caja");
        caja.innerHTML = "<b>dentro del DIV</b>";
    }
</script>
<div id="caja">Hola Mundo</div>
<input type="button" value="Borrar" onclick="borrar()" />
<input type="button" value="Mostrar Texto" onclick="texto()" />
</body>
</html>
```

Cambiar el aspecto del objeto cuando está seleccionado

223.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <style>
        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>
```

Funciona también así:

224.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <style>
        input {
            background-color: lightgreen;
            border: 1px solid gray;
        }

        input.enfoco {
            background-color: lightblue;
            border: 3px solid green;
        }
    </style>
</head>
<body>
<script>
    function consiguefoco(entrada) {
        entrada.className = 'enfoco';
    }

    function fueradelfoco(entrada) {
        entrada.className = '';
    }
</script>
<input type="text" id="DatoA" onfocus="consiguefoco(this)" onblur="fueradelfoco(this)" />
<input type="text" id="DatoB" onfocus="consiguefoco(this)" onblur="fueradelfoco(this)" />
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <style>
        input.error{ border: 1px solid red; background-color: yellow; }
    </style>
</head>
<body>
<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>
<label>Valor: <input type="text" id="Numero" onblur="validanumero(this)" /></label>
<label> Dato: <input type="text" id="Dato"/></label>
</body>
</html>
```

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

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

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

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Obtiene la lista de objetos tipo <p>
    //y muestra el valor que tiene cada uno
    function ChequeaObjetos() {
        let arreglo = document.getElementsByTagName("p");
        for (let cont = 0; cont < arreglo.length; cont++)
            alert("Contenido : " + arreglo[cont].innerHTML);
    }
</script>
<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>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Obtiene la lista de objetos tipo <p>
    //y cambia la apariencia de cada uno
    function CambiaObjetos() {
        let arreglo = document.getElementsByTagName("p");
        for (let 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>
<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>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Muestra la ubicación URL de la página
    function DarUbicacion() {
        let ubicacion = document.URL;
        alert("URL completa es: " + ubicacion);
    }
</script>
<input type="button" onclick="DarUbicacion()" value="¿Dónde estoy?">
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
    <style>
        .unestilo {
            color: blue;
            font-family: impact;
            font-size: 50px;
        }
    </style>
</head>
<body>
<script>
    //Aplica un atributo al primer <h1>
    function AplicarAtributo() {
        let objetoH1 = document.getElementsByTagName("H1")[0];
        let Atributo = document.createAttribute("class");
        Atributo.value = "unestilo";
        objetoH1.setAttributeNode(Atributo);
    }
</script>
<h1>Esta es una prueba</h1>
<h1>Segundo texto</h1>
<button onclick="AplicarAtributo()">Aplicar Atributo</button>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Muestra que codificación tiene la página
    function QueCodificacionTiene() {
        let codificacion = document.inputEncoding;
        alert("Codificación es: " + codificacion);
    }
</script>
<input type="button" onclick="QueCodificacionTiene()" value="Mi codificación">
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Lee el título de la página
    function QueTituloTiene() {
        let titulo = document.title;
        alert("Título es: " + titulo);
    }
</script>
<input type="button" onclick="QueTituloTiene()" value="Mi título">
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</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>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Muestra la fecha de modificación del documento
    function Modificacion() {
        let fecha = document.lastModified;
        alert("Este documento fue modificado: " + fecha);
    }
</script>
<input type="button" onclick="Modificacion()" value="¿Última modificación?">
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Muestra el tipo de documento
    function Tipo() {
        let tipodocumento = document.doctype.name;
        alert("Tipo de documento: " + tipodocumento);
    }
</script>
<input type="button" onclick="Tipo()" value="¿Tipo de documento?">
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Muestra el dominio donde está el documento
    function DominioDocumento() {
        let dominio = document.domain;
        alert("Dominio: " + dominio);
    }
</script>
<input type="button" onclick="DominioDocumento()" value="¿Dominio?">
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Muestra el estado del documento
    function Estado() {
        let estadodocumento = document.readyState;
        alert("Estado del documento: " + estadodocumento);
    }
</script>
<input type="button" onclick="Estado()" value="¿Está listo?">
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Muestra las dimensiones del documento
    //Nota: Al agregar más líneas con <p> aumenta la altura
    function Dimensiones() {
        let altura = document.body.clientHeight;
        let ancho = document.body.clientWidth;
        alert("Altura: " + altura + " Ancho: " + ancho);
    }
</script>
<input type="button" onclick="Dimensiones()" value="Tamaños">
<p>Prueba</p>
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Cuenta el número de hipervínculos que hay en el documento
    function TotalEnlaces() {
        alert("Enlaces: " + document.links.length);
    }
</script>
<a href="http://darwin.50webs.com">Mi página web</a><br>
<a href="https://github.com/ramsoftware/LibroRedNeuronal2020">Libro de Redes Neuronales</a><br>
<input type="button" onclick="TotalEnlaces()" value="¿Enlaces?">
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Muestra cada hipervínculo que hay en el documento
    function MuestraEnlaces() {
        for (let cont = 0; cont < document.links.length; cont++)
            alert("Enlace: " + document.links[cont].href);
    }
</script>
</body>
<a href="http://darwin.50webs.com">Mi página web</a><br>
<a href="https://github.com/ramsoftware/Evaluador3">Evaluador de expresiones</a><br>
<input type="button" onclick="MuestraEnlaces()" value="¿Enlaces?">
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Crea controles (botones) en tiempo de ejecución
    function CreaBotones() {
        //Genera un botón
        let boton = document.createElement("button");

        //Genera un texto
        let texto = document.createTextNode("Presionar");

        //Agrega el texto al botón
        boton.appendChild(texto);

        //Agrega el botón al <body>
        document.body.appendChild(boton);
    }
</script>
<input type="button" onclick="CreaBotones()" value="Generar Botones">
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Crea etiquetas en tiempo de ejecución
    function CreaEtiquetas() {
        //Genera una etiqueta
        let etiqueta = document.createElement("H1");

        //Genera un texto
        let 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>
<input type="button" onclick="CreaEtiquetas()" value="Generar Etiquetas">
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Crea etiquetas en tiempo de ejecución
    function CreaEtiquetas() {
        let etiqueta = document.createElement("H1");
        let 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() {
        let etiquetas = document.getElementsByTagName("H1");
        alert("Total etiquetas: " + etiquetas.length);
    }
</script>
<input type="button" onclick="CreaEtiquetas()" value="Generar Etiquetas">
<input type="button" onclick="TotalEtiquetas()" value="Total Etiquetas">
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Trae el número de elementos con un determinado "name"
    function TotalElementos() {
        let elementosNombre = document.getElementsByName("prueba");
        alert("Número de elementos: " + elementosNombre.length);
    }
</script>
<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>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Trae el número de formularios
    function TotalFormularios() {
        alert("Total formularios: " + document.forms.length);
    }
</script>
<form name="Form1"></form>
<form name="Form2"></form>
<form></form>
<input type="button" onclick="TotalFormularios()" value="¿Total Formularios?">
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Trae el total de imágenes en el documento
    function TotalImagenes() {
        alert("Total imágenes: " + document.images.length);
    }
</script>
<br>
<input type="button" onclick="TotalImagenes()" value="¿Total imágenes?">
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Describe los objetos de la página
    function Atributos() {
        let 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>
<p id="identifica" name="textual">Esto es una prueba</p>
<input type="button" onclick="Atributos()" value="¿Atributos del texto?">
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<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>
<input type="button" value="Abrir ventana" onclick="abrirventana()" />
</body>
</html>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Abre determinadas ventanas apuntando a diferentes direcciones
    let ventana = 1;
    let direccion = "https://github.com/ramsoftware";
    function abrirventana() {
        switch (ventana) {
            case 1:
                direccion = "http://darwin.50webs.com";
                break;
            case 2:
                direccion = "http://es.wikipedia.org";
                break;
            case 3:
                direccion = "https://www.youtube.com/@RafaelMorenoP";
                break;
        }
        ventana++;
        if (ventana === 4) ventana = 1;
        window.open(direccion, "_blank", "width=400,height=300");
    }
</script>
<input type="button" value="Abrir ventana" onclick="abrirventana()"/>
</body>
</html>
```

Un formulario que ejecuta un algoritmo local al enviar la información

El programa lee los datos que ha digitado el usuario, ejecuta y muestra en la misma página los resultados.

252.html

```
<!DOCTYPE HTML><html lang="es"><head><title>Capacitándose en JavaScript</title></head><body><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() {
    let cuenta = 0;
    let cadena = "";
    let limiteminimo = parseInt(document.getElementById("limitemin").value);
    let limitemaximo = parseInt(document.getElementById("limitemax").value);
    for (let numero = limiteminimo; numero <= limitemaximo; numero++) {
        if ((numero % 100) % 7 === 0) {
            cuenta++;
            cadena += numero + ", ";
        }
    }
    document.getElementById("resultado").value = cuenta;
    document.getElementById("datos").value = cadena;
}
</script>
<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>
```

Una segunda versión del programa modificando el uso del "submit"

253.html

```
</html><!DOCTYPE HTML><html lang="es"><head><title>Capacitándose en JavaScript</title></head><body>
<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() {
    let cuenta = 0;
    let cadena = "";
    let limiteminimo = parseInt(document.getElementById("limitemin").value);
    let limitemaximo = parseInt(document.getElementById("limitemax").value);
    for (let numero = limiteminimo; numero <= limitemaximo; numero++) {
        if ((numero % 100) % 7 === 0) {
            cuenta++;
            cadena += numero + ", ";
        }
    }
    document.getElementById("resultado").value = cuenta;
    document.getElementById("datos").value = cadena;
}
</script>
<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>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Valida un correo con una expresión regular */
    function validacorreo() {
        let correo = document.getElementById("direccion").value;
        let exprReg = /^(([^<>()[]\.\,\;\:\s@"]+(\[^<>()[]\.\,\;\:\s@"]+)*\.)|(\".+\")@\(([0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3})|(([a-zA-Z\-\-0-9]+\.)+[a-zA-Z]{2,}))$/;
        let 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>
<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>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Valida una URL con expresiones regulares */
    function validaURL() {
        let url = document.getElementById("direccion").value;
        let re = /^(ht|f)tp(s?)\:\/\/[0-9a-zA-Z]([-.\\w]*[0-9a-zA-Z])*(:(0-9)*)*(\/?) ([a-zA-Z0-9]-
        \.\\?\\,\\'\\/\\\\+&%\$#_]*\$)/;
        let resultado = re.test(url);
        if (resultado)
            document.getElementById("mensaje").value = "La URL es correcta";
        else
            document.getElementById("mensaje").value = "Es errónea la URL";
    }
</script>
<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>
```

Gráficos

Con HTML5 tenemos 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:

256.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</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:

257.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="miliendo" width="200" height="200" style="border:1px solid #000000;"></canvas>
<script>
    //Funciones para hacer gráficos
</script>
</body>
</html>
```



Ilustración 61:Lienzo ("canvas") visible usando un borde

Dibujar líneas

Usamos entonces instrucciones de JavaScript para hacer los gráficos. Se dibuja una línea.

258.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
    //Funciones para hacer gráficos

    //Captura el lienzo
    let lienzo = document.getElementById('milienzo');

    //Captura el contexto (siempre es en 2d)
    let 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 62: Línea dibujada

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="miliendo" width="400" height="400"></canvas>
<script>
    //Funciones para hacer gráficos

    //Captura el lienzo
    let lienzo = document.getElementById('miliendo');

    //Captura el contexto (siempre es en 2d)
    let 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 63: Línea gruesa

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
    //Funciones para hacer gráficos

    //Captura el lienzo
    let lienzo = document.getElementById('milienzo');

    //Captura el contexto (siempre es en 2d)
    let 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 64: Color a la línea

Color RGB

El color puede ser expresado en RGB (Red, Green Blue). Conociendo los valores de esos colores primarios podemos combinar.

261.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="miliendo" width="400" height="400"></canvas>
<script>
    //Funciones para hacer gráficos

    //Captura el lienzo
    let lienzo = document.getElementById('miliendo');

    //Captura el contexto (siempre es en 2d)
    let 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 65: Línea con color modificando los parámetros RGB

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
    //Funciones para hacer gráficos

    //Captura el lienzo
    let lienzo = document.getElementById('milienzo');

    //Captura el contexto (siempre es en 2d)
    let 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 66: Las líneas finalizan redondeadas en ambos extremos usando "round"

Dibujar varias líneas

Requiere llamar tres veces la instrucción beginPath(). Dos terminaciones distintas.

263.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="miliendo" width="400" height="400"></canvas>
<script>
    //Captura el lienzo
    let lienzo = document.getElementById('miliendo');

    //Captura el contexto (siempre es en 2d)
    let 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 línea

    //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 línea
</script>
</body>
</html>
```

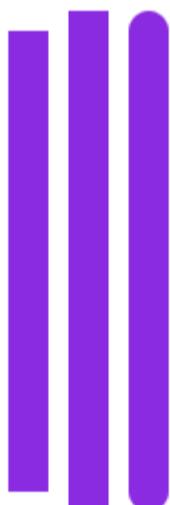


Ilustración 67:Tres líneas con la misma longitud, pero diferente cierre de extremo

Juntar líneas

Juntar dos líneas y que el punto de la unión tenga una característica (como ser redondeado)

264.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let 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>
```

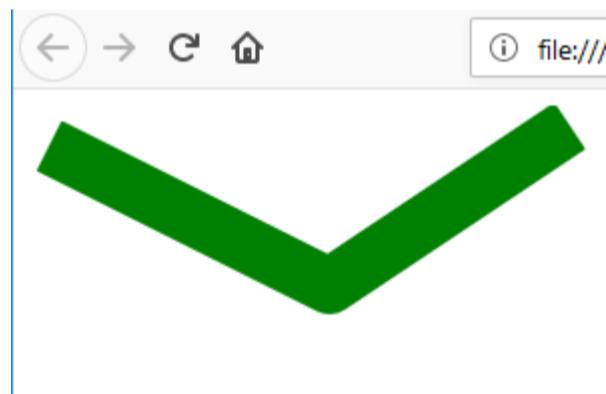


Ilustración 68: Dos líneas juntas con un empalme redondeado

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let contexto = lienzo.getContext('2d');

    contexto.lineWidth = 30; //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();

    //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>
```

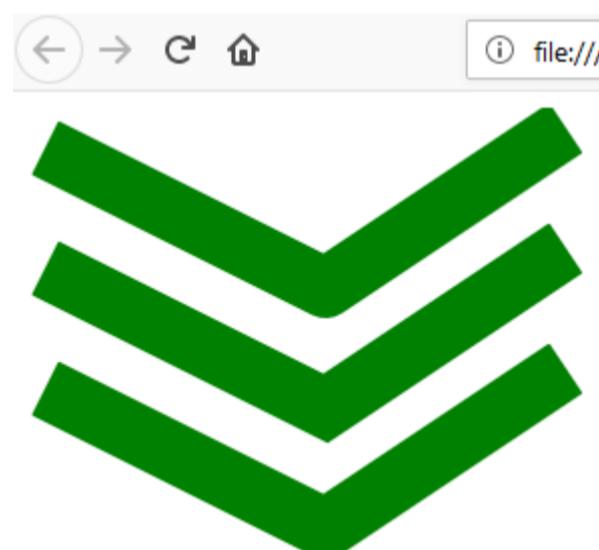


Ilustración 69: Diferentes formas de juntar dos líneas: "round", "miter", "bevel"

Varias líneas

Con tres líneas hacemos una figura y luego es cerrada (hacemos una línea donde termina la tercera línea y donde empezó la primera línea).

266.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="miliendo" width="400" height="400"></canvas>
<script>
    let lienzo = document.getElementById('miliendo');
    let 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>
```

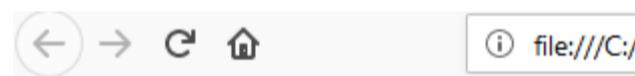


Ilustración 70: Una figura cerrada

Genera una figura cerrada y la rellena

267.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
  <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
  let lienzo = document.getElementById('milienzo');
  let 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 71: Figura cerrada y rellena de un color

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let contexto = lienzo.getContext('2d');

    //Dibuja varias líneas horizontales
    contexto.beginPath(); //Inicia el camino
    for (let coordY = 10; coordY <= 500; coordY += 10) {
        contexto.moveTo(5, coordY);
        contexto.lineTo(600, coordY);
    }
    contexto.stroke();
</script>
</body>
</html>
```

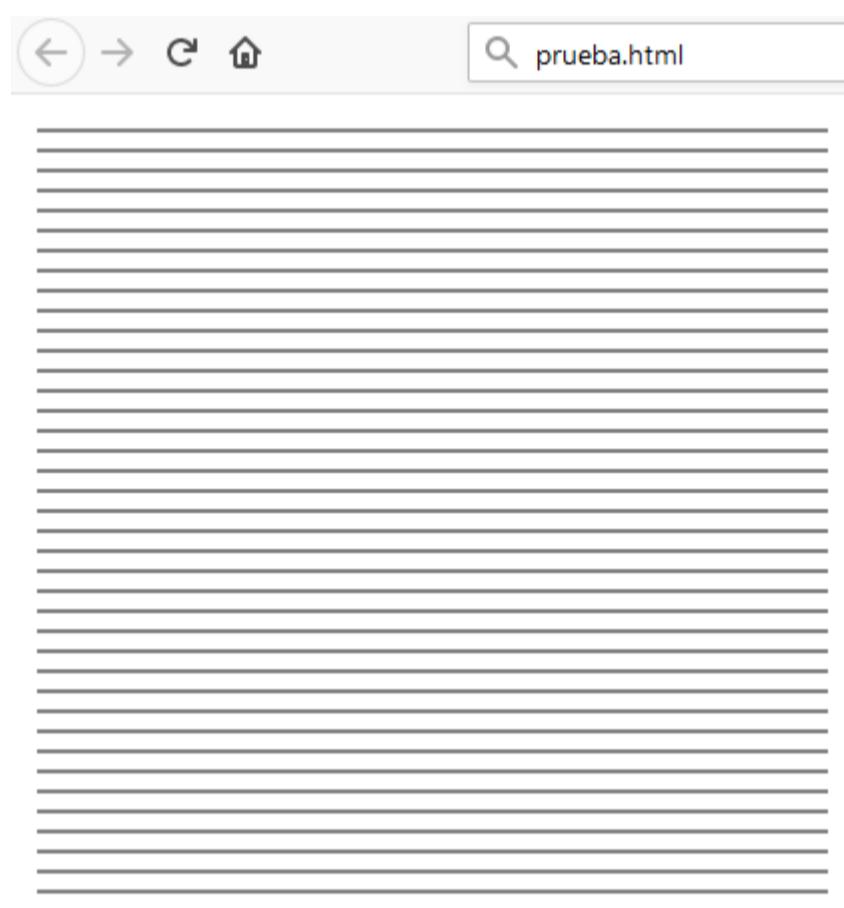


Ilustración 72: Líneas generadas por un ciclo

Dibujar varias líneas verticales con un ciclo

269.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
  <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
  let lienzo = document.getElementById('milienzo');
  let contexto = lienzo.getContext('2d');

  //Dibuja varias líneas verticales
  contexto.beginPath(); //Inicia el camino
  for (let coordX = 10; coordX <= 500; coordX += 50) {
    contexto.moveTo(coordX, 10);
    contexto.lineTo(coordX, 400);
  }
  contexto.stroke();
</script>
</body>
</html>
```

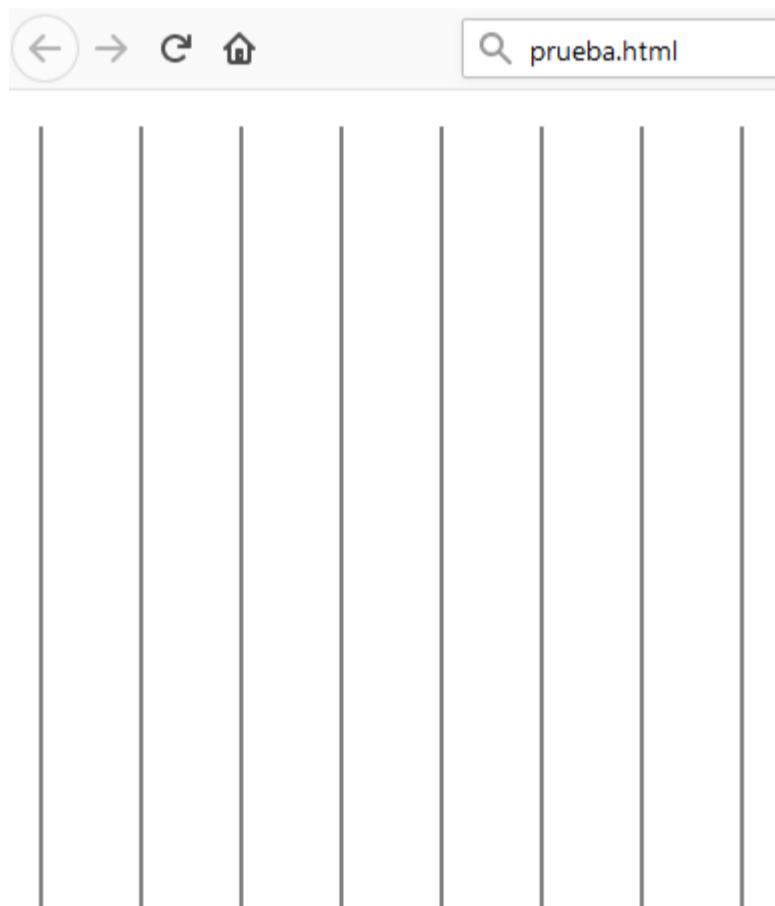


Ilustración 73: Líneas verticales dibujadas con un ciclo

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="600" height="600"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let contexto = lienzo.getContext('2d');

    contexto.beginPath(); //Inicia el camino

    //Ilusión de curva con líneas
    for (let mover = 0; mover <= 600; mover += 10) {
        contexto.moveTo(0, mover);
        contexto.lineTo(mover, 600);
    }
    contexto.stroke();
</script>
</body>
</html>
```

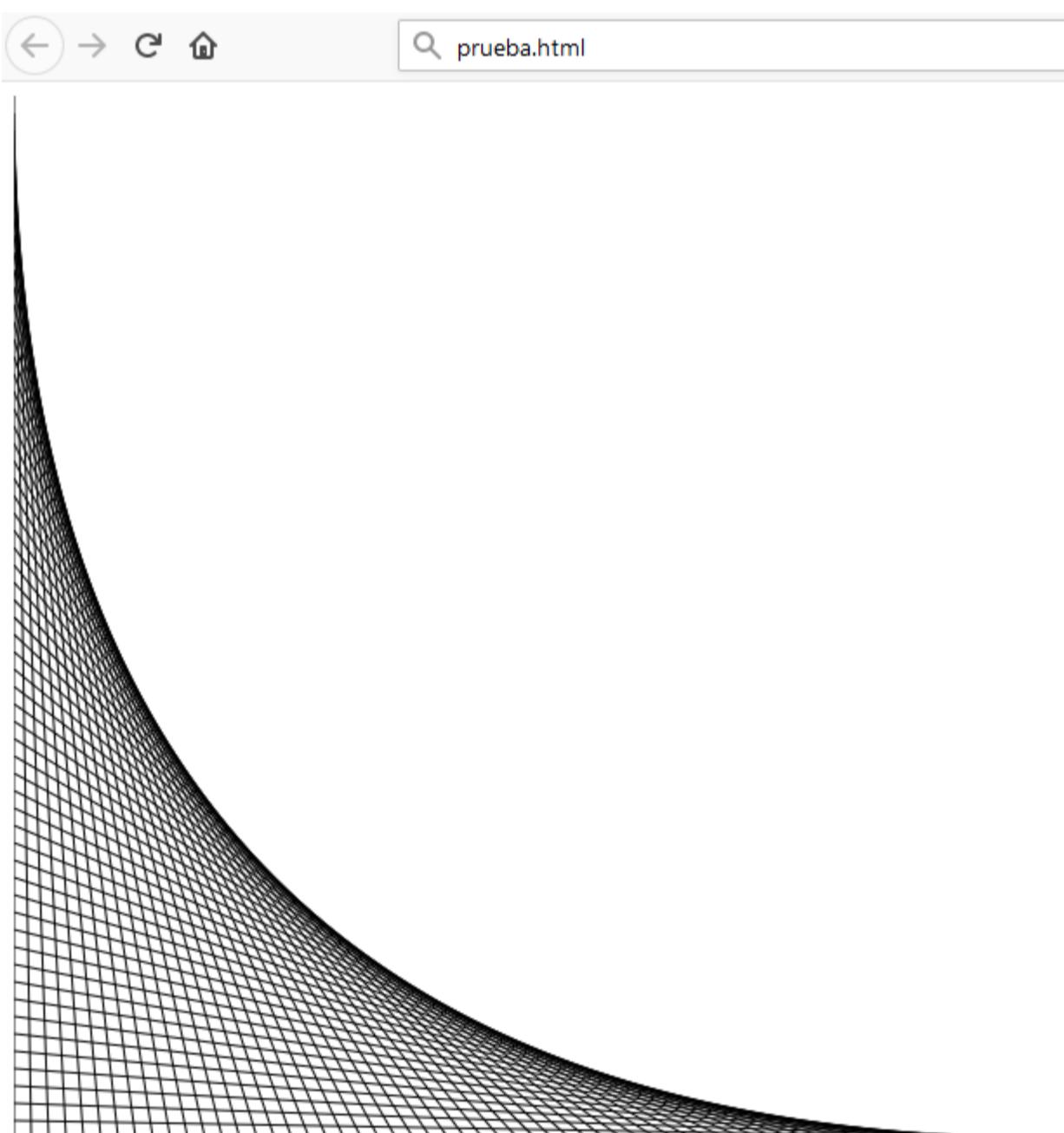


Ilustración 74: Ilusión de curva usando líneas rectas

"Curvas" con líneas

```
<!DOCTYPE HTML>
<html lang="es">
<head>
  <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="miliendo" width="600" height="600"></canvas>
<script>
  let lienzo = document.getElementById('miliendo');
  let contexto = lienzo.getContext('2d');

  contexto.beginPath(); //Inicia el camino

  //Curvas usando líneas
  for (let 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>
```

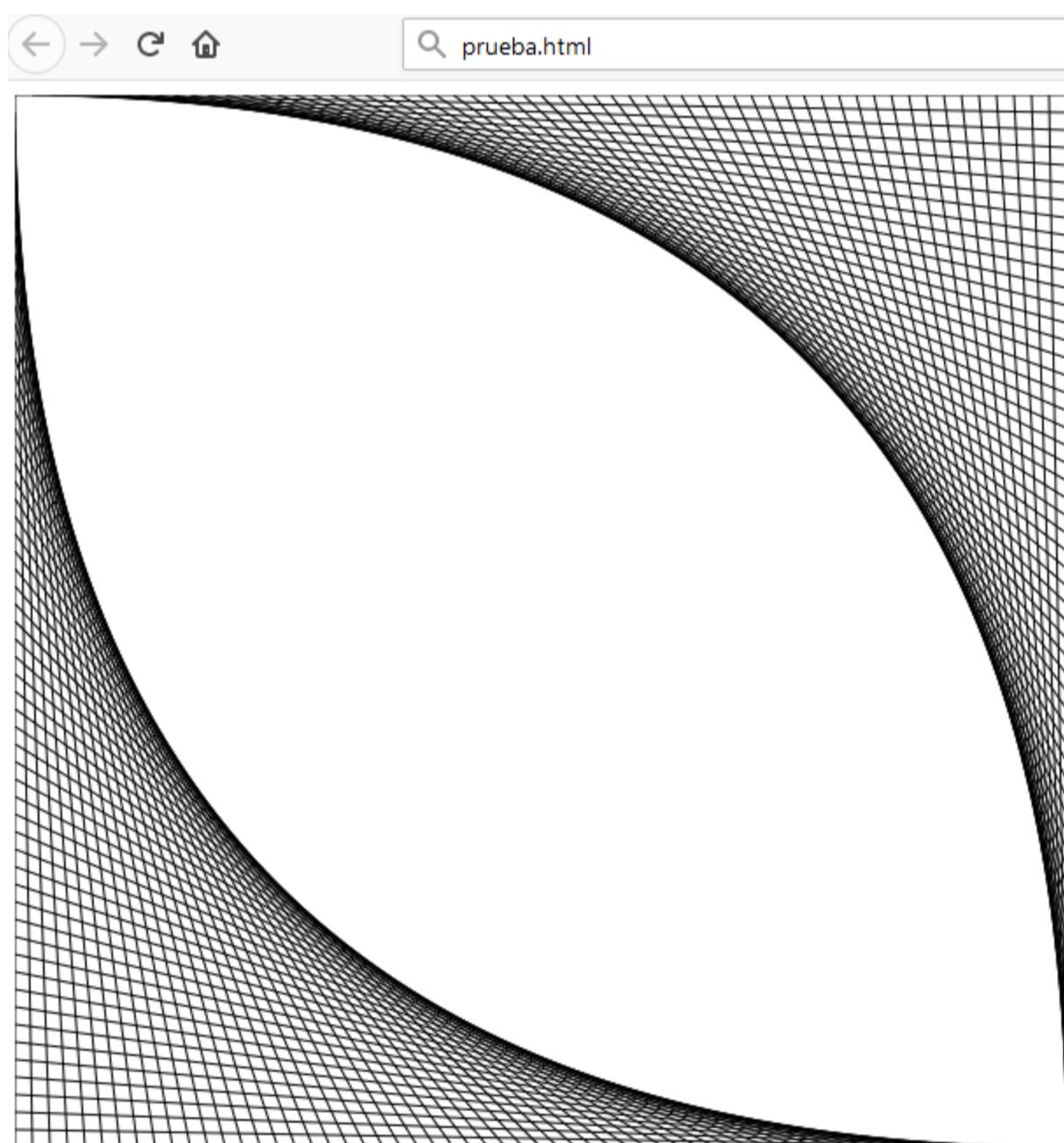


Ilustración 75: Ilusión de dos curvas usando líneas rectas

Dibujar círculos

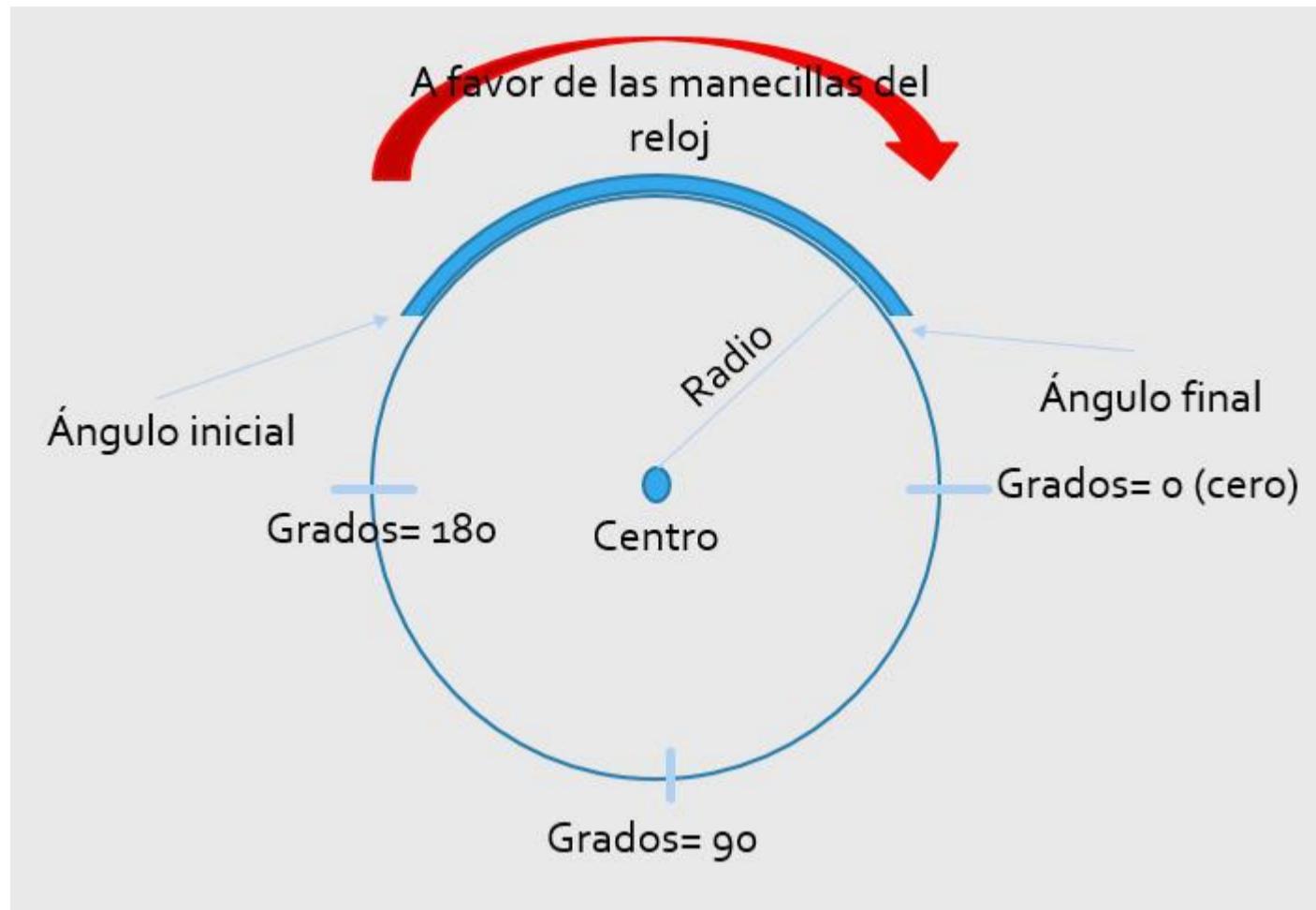


Ilustración 76: La manera en que se dibujará el círculo

272.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="miliendo" width="300" height="300"></canvas>
<script>
    let lienzo = document.getElementById('miliendo');
    let contexto = lienzo.getContext('2d');

    let posX = lienzo.width / 2; //Centro del canvas en X
    let posY = lienzo.height / 2; //Centro del canvas en Y
    let Radio = 100; //Radio del círculo
    let AnguloInicio = 0; //Angulo inicial (0 grados) en radianes
    let AnguloFinal = 360 * Math.PI / 180; //Angulo final (360 grados) en radianes
    let DibujaContraReloj = false;

    contexto.beginPath();
    contexto.arc(posX, posY, Radio, AnguloInicio, AnguloFinal, DibujaContraReloj);
    contexto.lineWidth = 10; //Ancho de la linea
    contexto.strokeStyle = 'green'; //Color
    contexto.stroke();
</script>
</body>
</html>
```

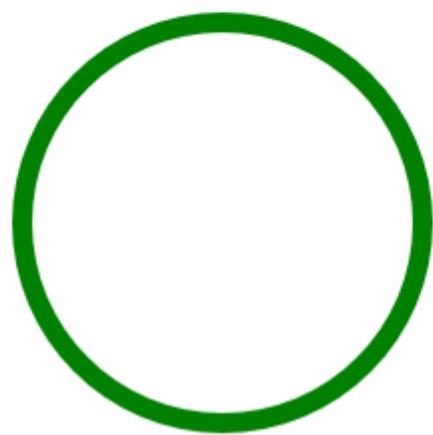


Ilustración 77: Círculo

Dibujar curvas

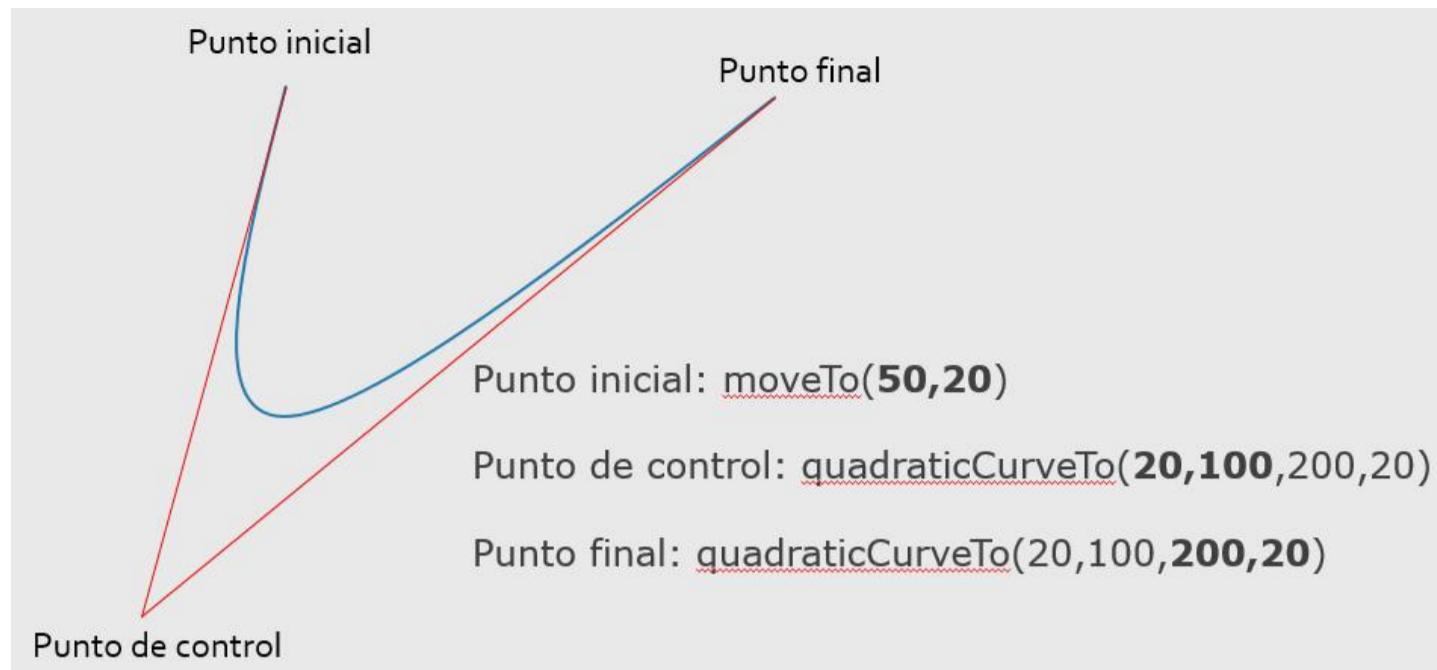


Ilustración 78: Forma en que se dibujarán las curvas

273.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="miliendo" width="300" height="300"></canvas>
<script>
    let lienzo = document.getElementById('miliendo');
    let 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 linea
    contexto.strokeStyle = 'green'; //Color
    contexto.stroke();
</script>
</body>
</html>
```

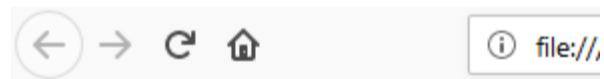
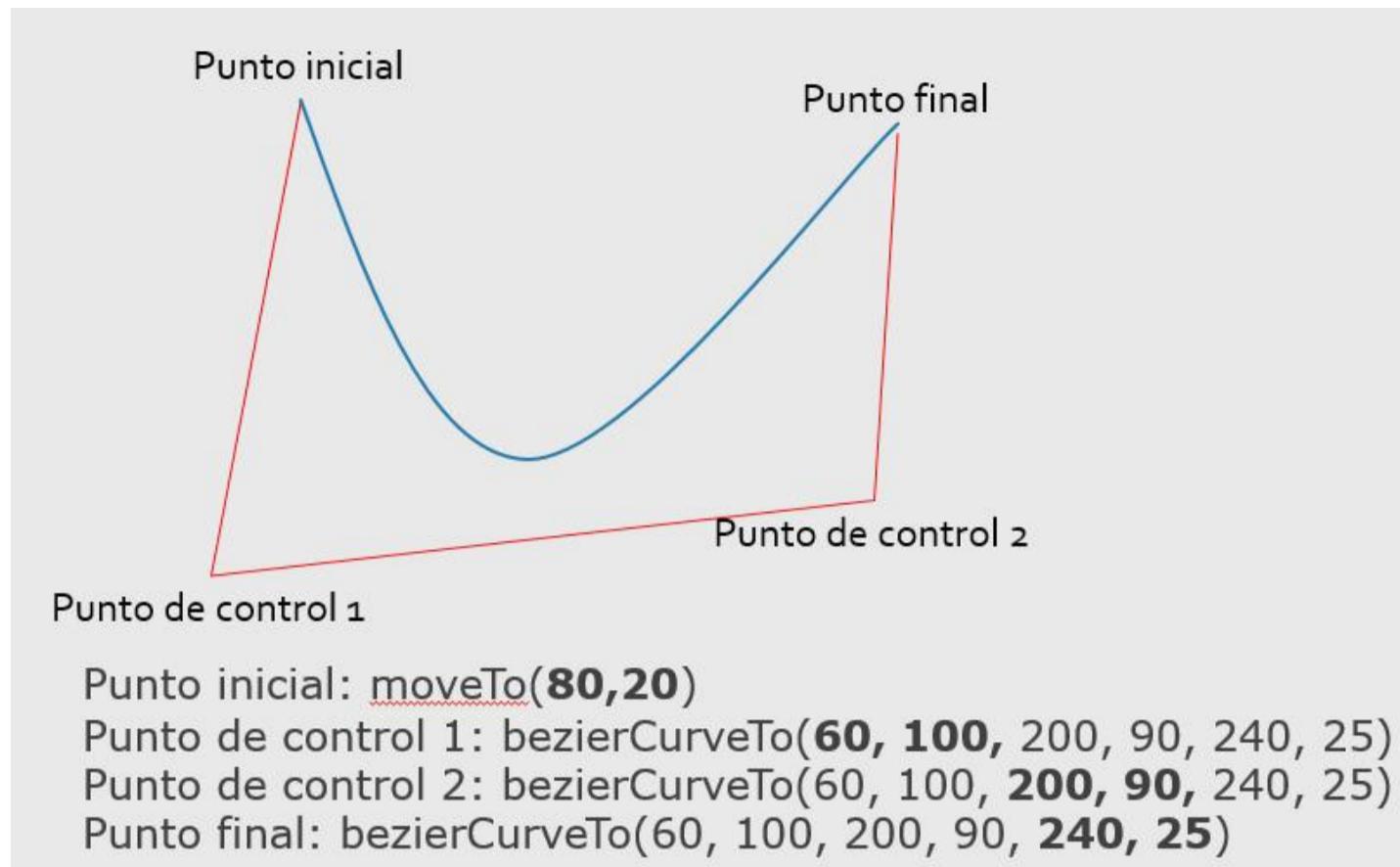


Ilustración 79: Curva

Curva de Bézier



274.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
  <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="miliendo" width="300" height="300"></canvas>
<script>
  let lienzo = document.getElementById('miliendo');
  let 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>
```

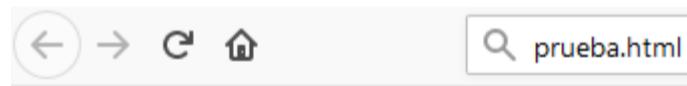


Ilustración 80: Curva Bézier

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="300" height="300"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let 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>
```

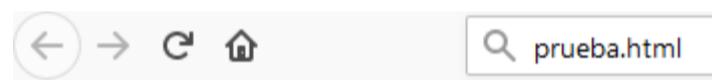


Ilustración 81: Combinando curvas

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.

276.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let contexto = lienzo.getContext('2d');

    //Hace un rectángulo
    contexto.rect(0, 0, lienzo.width, lienzo.height);

    //Gradiente de cambio de color
    let 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>
```

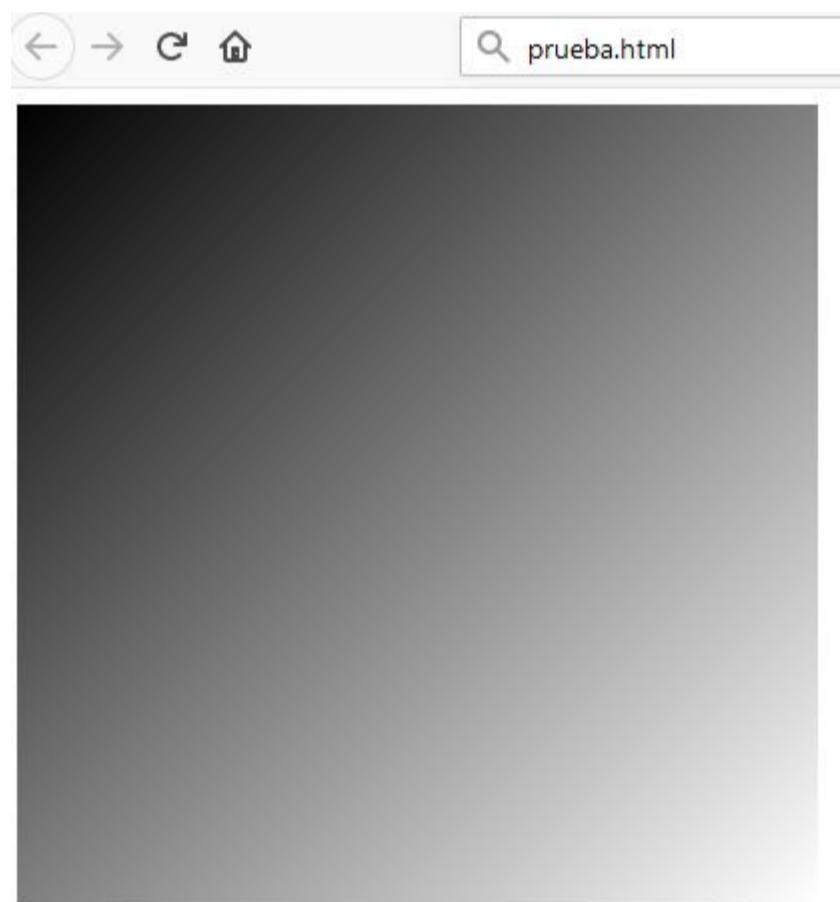


Ilustración 82: Variación del color

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let 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. */
    let 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>
```

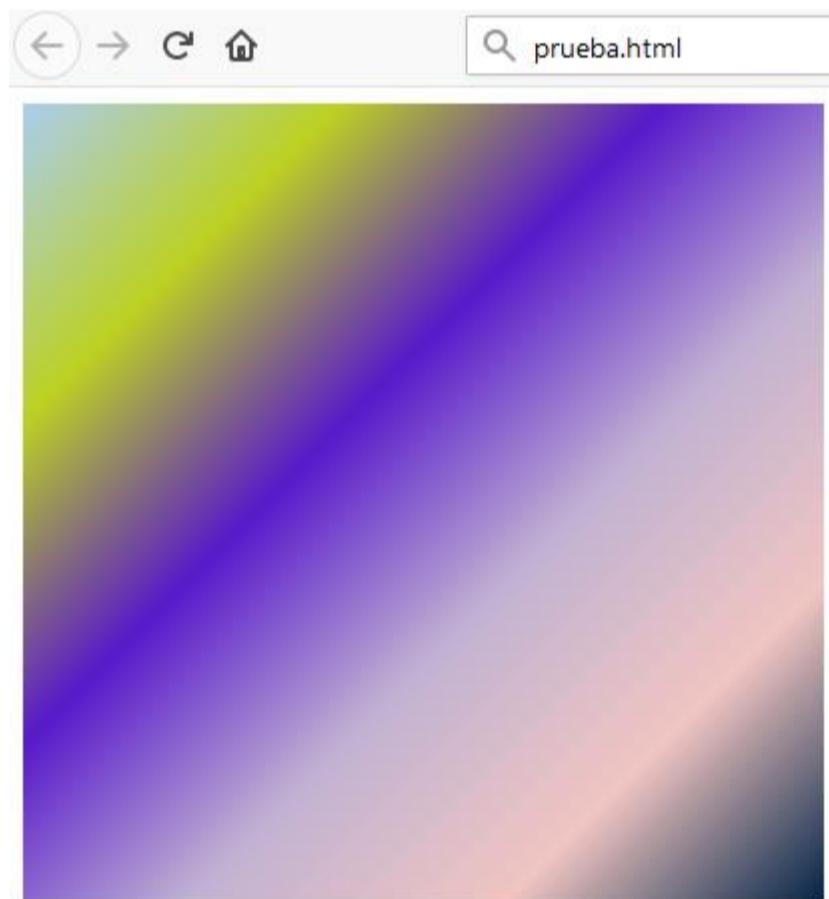


Ilustración 83: Variación de varios colores en forma oblicua

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let contexto = lienzo.getContext('2d');

    //Hace un rectángulo
    contexto.rect(0, 0, lienzo.width, lienzo.height);

    /* Crea un gradiente de color de variación vertical */
    let 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>
```

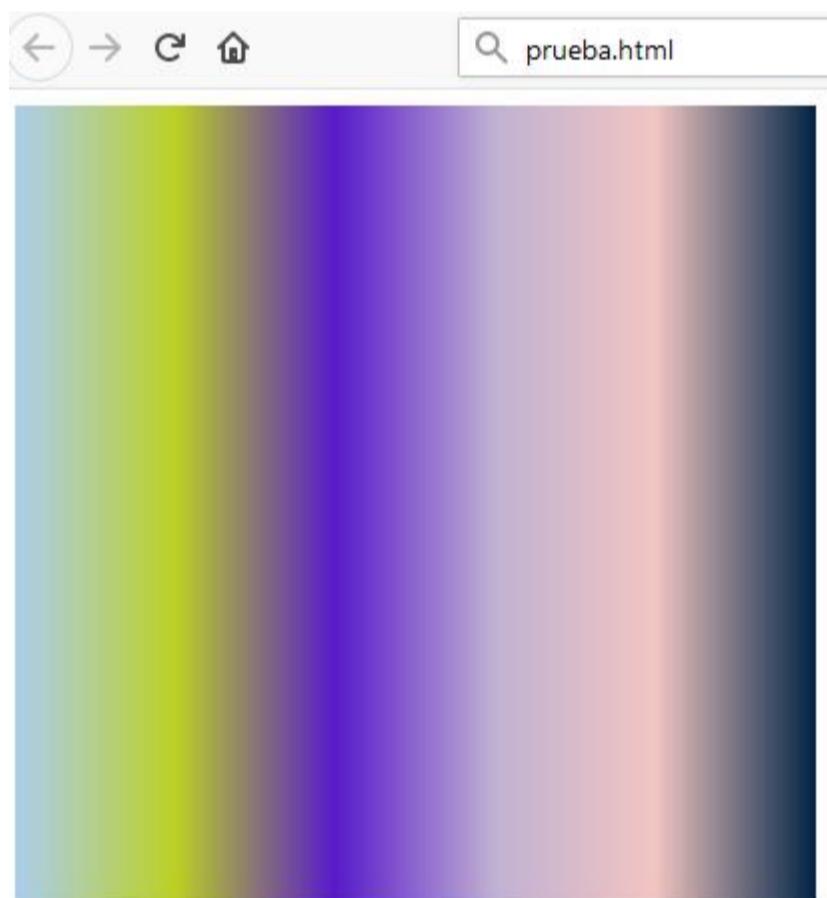


Ilustración 84: Variación de color en forma vertical

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let contexto = lienzo.getContext('2d');

    //Hace un rectángulo
    contexto.rect(0, 0, lienzo.width, lienzo.height);

    /* Crea un gradiente de color de variación horizontal */
    let 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>
```

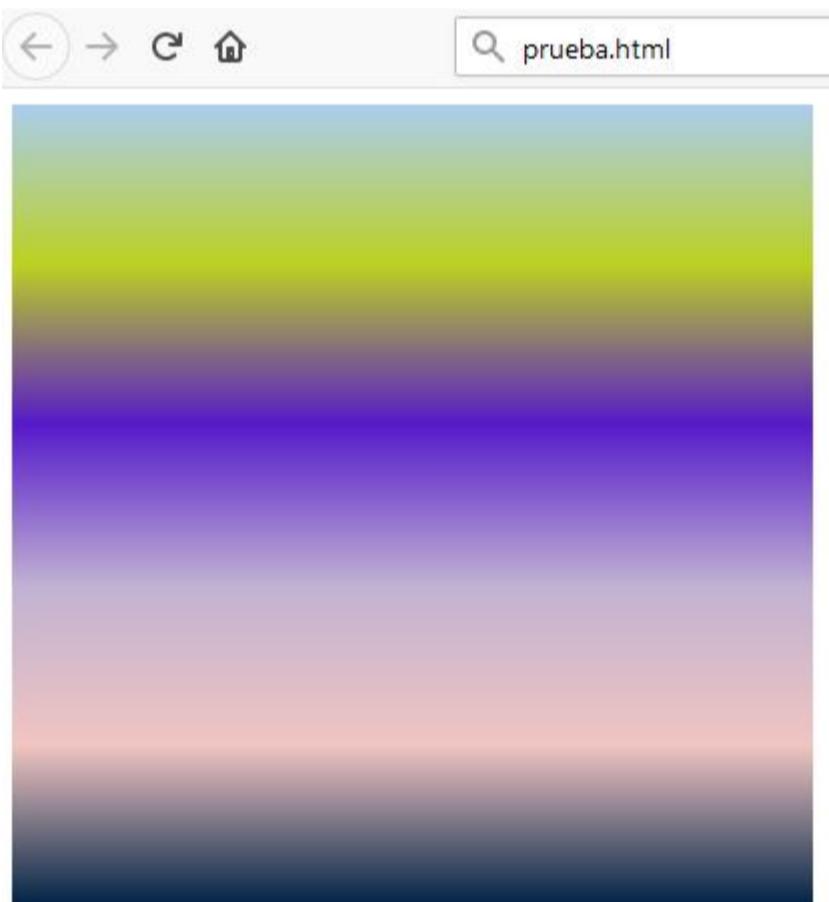


Ilustración 85: Variación de color horizontal

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let 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 */
    let 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>
```

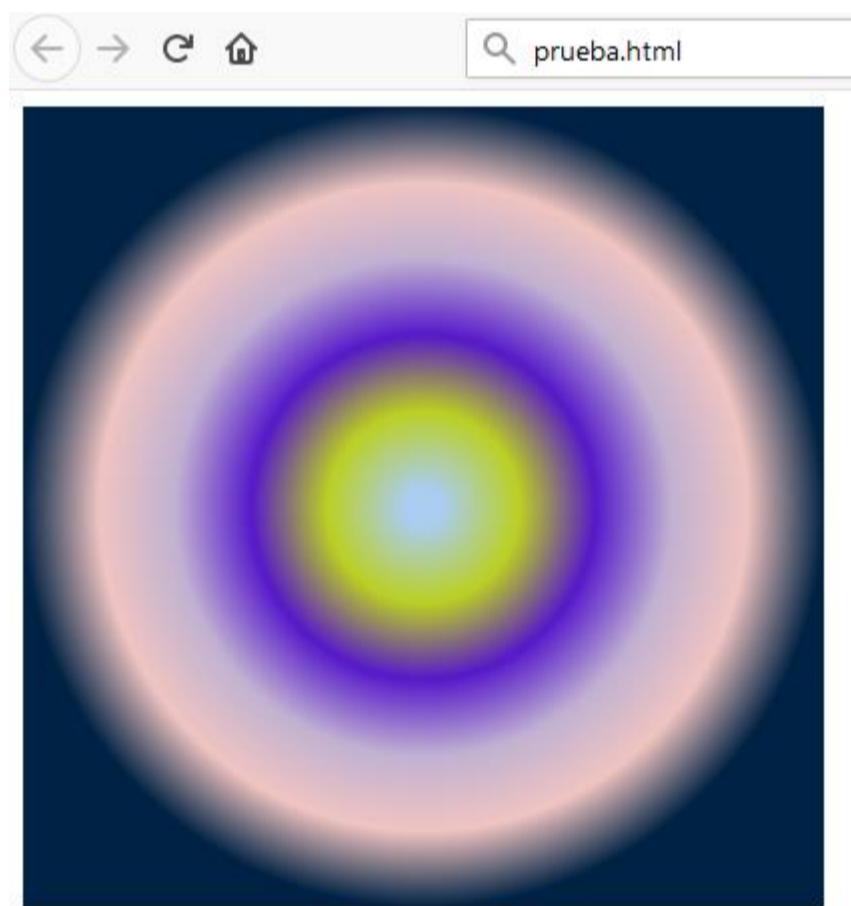


Ilustración 86: Variación en círculos

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="400" height="400"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let 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 () {
        contexto.fillStyle = contexto.createPattern(imagen, 'repeat');
        contexto.fill(); //Rellena
    }

    /* Probemos con esto también:
        let patron = contexto.createPattern(imagen, 'repeat-x');
        let patron = contexto.createPattern(imagen, 'repeat-y');
        let patron = contexto.createPattern(imagen, 'no-repeat'); */
</script>
</body>
</html>
```



Ilustración 87: La imagen rellena el rectángulo repitiéndose

Variar posición y tamaño de la imagen

Muestra la imagen, se puede variar la posición en el lienzo y su tamaño en ancho y alto

282.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="miliendo" width="400" height="400"></canvas>
<script>
    let lienzo = document.getElementById('miliendo');
    let 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>
```

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="miLienzo" width="595" height="420"></canvas>
<script>
    let 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) {
        let lienzo = document.getElementById('miLienzo');
        let contexto = lienzo.getContext('2d');
        contexto.drawImage(Imagen, 0, 0); //Dibuja la imagen original en 0,0

        //Trae los datos de esa imagen
        let datoImagen = contexto.getImageData(0, 0, Imagen.width, Imagen.height);
        let Datos = datoImagen.data;

        //Se va de pixel en pixel y lo convierte a escala de grises
        for (let i = 0; i < Datos.length; i += 4) {
            let 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 88: Imagen original

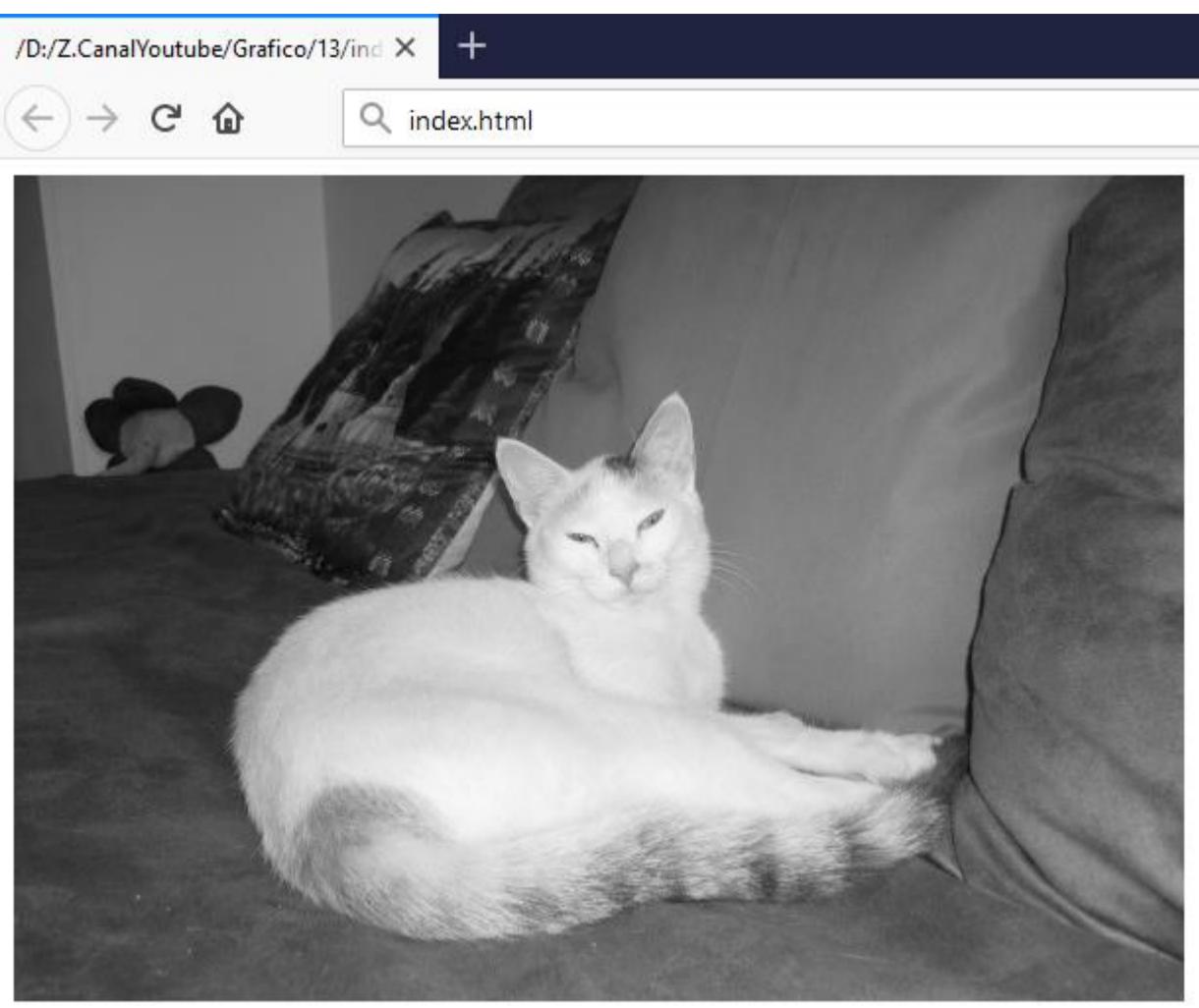


Ilustración 89: Imagen en escala de grises

Dibujar letras

Se pueden imprimir textos como imágenes.

284.html

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="700" height="400"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let 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

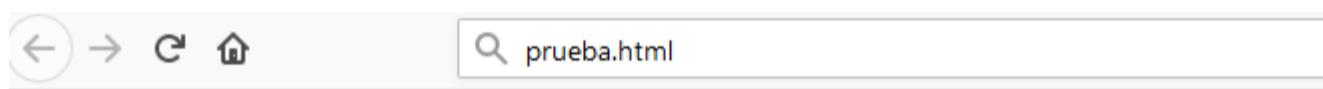
Ilustración 90: Dibuja un texto

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="700" height="400"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let 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 91: Dibuja un texto con relleno de color

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="700" height="400"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let contexto = lienzo.getContext('2d');

    //Tipo de letra, tamaño, fuente
    contexto.font = '50pt Verdana';

    //Ancho de la linea
    contexto.lineWidth = 1;

    //Color de la linea
    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 92: Texto dibujado, se muestra el borde de cada letra

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="miliendo" width="700" height="400"></canvas>
<script>
    let lienzo = document.getElementById('miliendo');
    let 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>
```

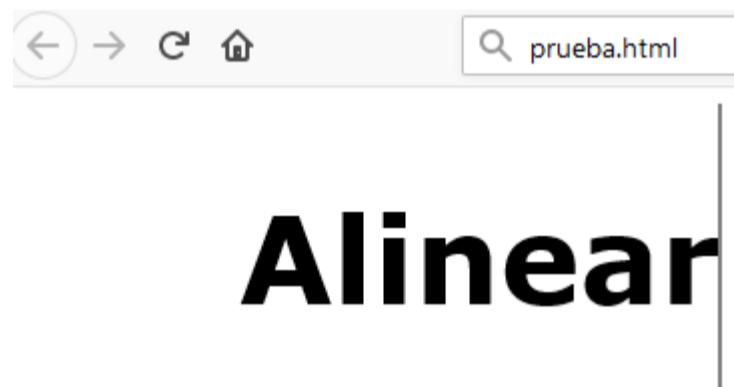


Ilustración 93: Dibuja un texto alineado al final

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="700" height="400"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let contexto = lienzo.getContext('2d');

    //Tipo de letra, tamaño, fuente
    contexto.font = '80pt Courier';

    //Texto a imprimir, posición X, posición Y
    let texto = "Prueba";
    contexto.fillText(texto, 10, 100);

    //Tamaño del texto
    let tamano = contexto.measureText(texto);
    let ancho = tamano.width;
    alert(ancho);
</script>
</body>
</html>
```



Ilustración 94: Muestra el ancho del texto dibujado

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="700" height="400"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let 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 95: Sombra

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="700" height="400"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let 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>
```

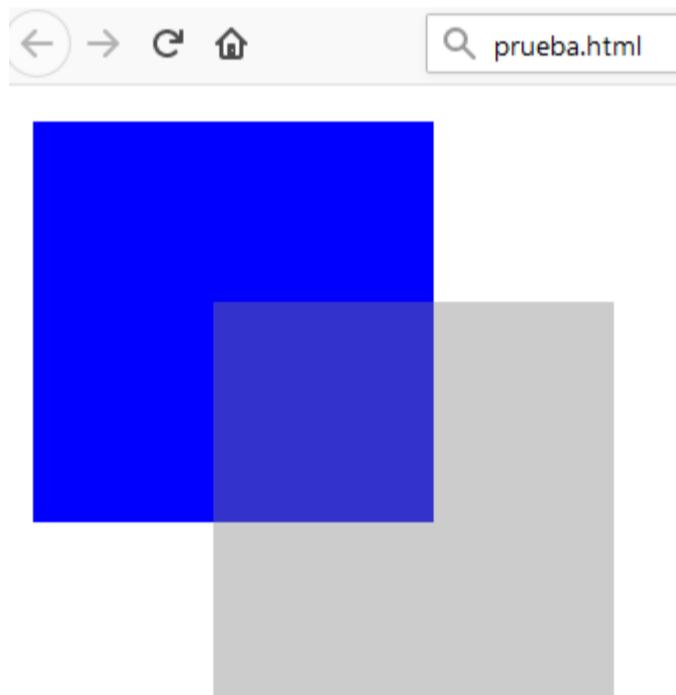


Ilustración 96: Rectángulo semitransparente

Librería Gráfica

Una forma más sencilla de trabajar con gráficos es utilizar la técnica "wrapper" (envoltorio) de las funciones gráficas nativas de JavaScript. A continuación, la librería gráfica:

LibreriaGrafica.js

```
/** Clase con funciones para hacer gráficos
 * @Author Rafael Alberto Moreno Parra
 * @Version 1.1 (07 de julio de 2025)
 */
class Grafica {
    /**
     * Captura el área gráfica (canvas)
     * @param {HTMLCanvasElement} Lienzo - El lienzo (id del canvas) de HTML5
     */
    constructor (Lienzo){
        this.Lienzo = Lienzo;
        this.Contexto = Lienzo.getContext('2d');
    }

    /**
     * Retorna el alto del área gráfica o lienzo
     * @return {number} Alto
     */
    LienzoAlto(){
        return this.Lienzo.height;
    }

    /**
     * Retorna el ancho del área gráfica o lienzo
     * @return {number} Ancho
     */
    LienzoAncho(){
        return this.Lienzo.width;
    }

    /**
     * Dibuja el perímetro del lienzo
     * @param {number} GrosorBorde - Grosor de la línea del perímetro
     * @param {string} ColorBorde - Color expresado en hexadecimal que tendrá la línea del perímetro
     */
    LienzoBorde(GrosorBorde, ColorBorde){
        this.Rectangulo(0, 0, this.Lienzo.width, this.Lienzo.height, GrosorBorde, ColorBorde);
    }

    /**
     * Poner un color de fondo al lienzo
     * @param {string} ColorFondo - Color expresado en hexadecimal que tendrá el fondo del lienzo
     */
    LienzoFondo(ColorFondo){
        this.RectanguloRelleno(0, 0, this.Lienzo.width, this.Lienzo.height, ColorFondo);
    }

    /**
     * Fondo y borde del lienzo
     * @param {string} ColorFondo - Color expresado en hexadecimal que tendrá el fondo del lienzo
     * @param {number} GrosorBorde - Grosor de la línea del perímetro
     * @param {string} ColorBorde - Color expresado en hexadecimal que tendrá la línea del perímetro
     */
    LienzoFondoBorde(ColorFondo, GrosorBorde, ColorBorde){
        this.LienzoFondo(ColorFondo);
        this.LienzoBorde(GrosorBorde, ColorBorde);
    }

    /**
     * Dibuja una línea recta
     * @param {number} Xinicio - Valor X de la coordenada inicial
     * @param {number} Yinicio - Valor Y de la coordenada inicial
     * @param {number} Xfin - Valor X de la coordenada final
     * @param {number} Yfin - Valor Y de la coordenada final
     * @param {number} GrosorLinea - Grosor de la línea
     * @param {string} ColorLinea - Color expresado en hexadecimal que tendrá la línea
     */
    Linea(Xinicio, Yinicio, Xfin, Yfin, GrosorLinea, ColorLinea){
        this.Contexto.beginPath();
        this.Contexto.lineWidth = GrosorLinea;
        this.Contexto.strokeStyle = ColorLinea;
        this.Contexto.moveTo(Xinicio, Yinicio); //Mueve el cursor a la posición
        this.Contexto.lineTo(Xfin, Yfin); //Hace una línea
        this.Contexto.stroke(); //Hace visible la línea
        this.Contexto.closePath();
    }
}
```

```

/**
 * Dibuja las líneas que conectan varios puntos
 * @param {Punto[]} Puntos - Lista de objetos de la clase Punto
 * @param {number} GrosorLinea - Grosor de la línea
 * @param {string} ColorLinea - Color expresado en hexadecimal que tendrá la línea
 */
LineasPuntos(Puntos, GrosorLinea, ColorLinea) {
    this.Contexto.beginPath();
    this.Contexto.lineWidth = GrosorLinea;
    this.Contexto.strokeStyle = ColorLinea;
    this.Contexto.moveTo(Puntos[0].posX, Puntos[0].posY); //Punto inicial
    for(let cont = 1; cont < Puntos.length; cont++)
        this.Contexto.lineTo(Puntos[cont].posX, Puntos[cont].posY);
    this.Contexto.stroke(); //Hace visible la línea
    this.Contexto.closePath();
}

/**
 * Dibuja las líneas que hay en un listado
 * @param {Linea[]} LineaLst - Lista de objetos de la clase Linea
 */
LineasListas(LineaLst) {
    for(let cont = 0; cont < LineaLst.length; cont++)
        this.LineaObj(LineaLst[cont]);
}

/**
 * Dibuja una línea dado el objeto linea
 * @param {Linea} obj - Objeto Línea
 */
LineaObj(obj) {
    this.Linea(obj.posXini, obj.posYini, obj.posXfin, obj.posYfin, obj.Grosor, obj.Color);
}

/**
 * Dibuja un triángulo dada las coordenadas
 * @param {number} posX1 - Valor X de la primera coordenada
 * @param {number} posY1 - Valor Y de la primera coordenada
 * @param {number} posX2 - Valor X de la segunda coordenada
 * @param {number} posY2 - Valor Y de la segunda coordenada
 * @param {number} posX3 - Valor X de la tercera coordenada
 * @param {number} posY3 - Valor Y de la tercera coordenada
 * @param {number} GrosorBorde - Grosor del borde del triángulo
 * @param {string} ColorBorde - Color expresado en hexadecimal para el borde
 */
Triangulo(posX1, posY1, posX2, posY2, posX3, posY3, GrosorBorde, ColorBorde) {
    this.Contexto.beginPath();
    this.Contexto.lineWidth = GrosorBorde;
    this.Contexto.strokeStyle = ColorBorde;
    this.Contexto.moveTo(posX1, posY1); //Mueve el cursor a la posición
    this.Contexto.lineTo(posX2, posY2);
    this.Contexto.lineTo(posX3, posY3);
    this.Contexto.lineTo(posX1, posY1);
    this.Contexto.stroke(); //Hace visible el triángulo
    this.Contexto.closePath();
}

/**
 * Dibuja un triángulo relleno dada las coordenadas
 * @param {number} posX1 - Valor X de la primera coordenada
 * @param {number} posY1 - Valor Y de la primera coordenada
 * @param {number} posX2 - Valor X de la segunda coordenada
 * @param {number} posY2 - Valor Y de la segunda coordenada
 * @param {number} posX3 - Valor X de la tercera coordenada
 * @param {number} posY3 - Valor Y de la tercera coordenada
 * @param {string} ColorRelleno - Color expresado en hexadecimal para el relleno
 */
TrianguloRelleno(posX1, posY1, posX2, posY2, posX3, posY3, ColorRelleno) {
    this.Contexto.beginPath();
    this.Contexto.fillStyle = ColorRelleno;
    this.Contexto.moveTo(posX1, posY1); //Mueve el cursor a la posición
    this.Contexto.lineTo(posX2, posY2);
    this.Contexto.lineTo(posX3, posY3);
    this.Contexto.lineTo(posX1, posY1);
    this.Contexto.fill(); //Rellena con el color
    this.Contexto.closePath();
}

/**
 * Dibuja un triángulo relleno de un color y borde de otro color
 * @param {number} posX1 - Valor X de la primera coordenada
 * @param {number} posY1 - Valor Y de la primera coordenada
 * @param {number} posX2 - Valor X de la segunda coordenada
 * @param {number} posY2 - Valor Y de la segunda coordenada
 * @param {number} posX3 - Valor X de la tercera coordenada
 * @param {number} posY3 - Valor Y de la tercera coordenada
 */

```

```

* @param {string} ColorRelleno - Color expresado en hexadecimal para el relleno
* @param {number} GrosorBorde - Grosor del borde del triángulo
* @param {string} ColorBorde - Color expresado en hexadecimal para el borde
*/
TrianguloRellenoBorde(posX1, posY1, posX2, posY2, posX3, posY3, ColorRelleno, GrosorBorde, ColorBorde){
    this.TrianguloRelleno(posX1, posY1, posX2, posY2, posX3, posY3, ColorRelleno);
    this.Triangulo(posX1, posY1, posX2, posY2, posX3, posY3, GrosorBorde, ColorBorde);
}

/**
 * Dibuja una lista de triángulos
 * @param {Triangulo[]} TrianguloLst - Lista de objetos de la clase Triangulo
 */
TrianguloLista(TrianguloLst){
    for(let cont = 0; cont < TrianguloLst.length; cont++)
        this.TrianguloObj(TrianguloLst[cont]);
}

/**
 * Dibuja un triángulo dado el objeto triángulo
 * @param {Triangulo} obj - Objeto Triangulo
 */
TrianguloObj(obj){
    switch(obj.Tipo) {
        case 1: this.Triangulo(obj.posX1, obj.posY1, obj.posX2, obj.posY2, obj.posX3, obj.posY3,
obj.GrosorBorde, obj.ColorBorde); break;
        case 2: this.TrianguloRelleno(obj.posX1, obj.posY1, obj.posX2, obj.posY2, obj.posX3, obj.posY3,
obj.ColorRelleno); break;
        case 3: this.TrianguloRellenoBorde(obj.posX1, obj.posY1, obj.posX2, obj.posY2, obj.posX3, obj.posY3,
obj.ColorRelleno, obj.GrosorBorde, obj.ColorBorde); break;
    }
}

/**
 * Dibuja un rectángulo
 * @param {number} posX - Posición X de la coordenada superior izquierda del rectángulo
 * @param {number} posY - Posición Y de la coordenada superior izquierda del rectángulo
 * @param {number} Ancho - Ancho del rectángulo
 * @param {number} Alto - Alto del rectángulo
 * @param {number} GrosorBorde - Grosor del borde del rectángulo
 * @param {string} ColorBorde - Color expresado en hexadecimal para el borde
 */
Rectangulo(posX, posY, Ancho, Alto, GrosorBorde, ColorBorde){
    this.Contexto.beginPath();
    this.Contexto.lineWidth = GrosorBorde;
    this.Contexto.strokeStyle = ColorBorde;
    this.Contexto.rect(posX, posY, Ancho, Alto);
    this.Contexto.stroke(); //Hace visible el rectángulo
    this.Contexto.closePath();
}

/**
 * Dibuja un rectángulo relleno
 * @param {number} posX - Posición X de la coordenada superior izquierda del rectángulo
 * @param {number} posY - Posición Y de la coordenada superior izquierda del rectángulo
 * @param {number} Ancho - Ancho del rectángulo
 * @param {number} Alto - Alto del rectángulo
 * @param {string} ColorRelleno - Color expresado en hexadecimal para el relleno
 */
RectanguloRelleno(posX, posY, Ancho, Alto, ColorRelleno){
    this.Contexto.beginPath();
    this.Contexto.fillStyle = ColorRelleno;
    this.Contexto.fillRect(posX, posY, Ancho, Alto);
    this.Contexto.closePath();
}

/**
 * Dibuja un rectángulo relleno con borde
 * @param {number} posX - Posición X de la coordenada superior izquierda del rectángulo
 * @param {number} posY - Posición Y de la coordenada superior izquierda del rectángulo
 * @param {number} Ancho - Ancho del rectángulo
 * @param {number} Alto - Alto del rectángulo
 * @param {string} ColorRelleno - Color expresado en hexadecimal para el relleno
 * @param {number} GrosorBorde - Grosor del borde del rectángulo
 * @param {string} ColorBorde - Color expresado en hexadecimal para el borde
 */
RectanguloRellenoBorde(posX, posY, Ancho, Alto, ColorRelleno, GrosorBorde, ColorBorde){
    this.RectanguloRelleno(posX, posY, Ancho, Alto, ColorRelleno);
    this.Rectangulo(posX, posY, Ancho, Alto, GrosorBorde, ColorBorde);
}

/**
 * Dibuja una lista de rectángulos
 * @param {Rectangulo[]} RectanguloLst - Lista de objetos de la clase Rectangulo
 */
RectanguloLista(RectanguloLst){
    for(let cont = 0; cont < RectanguloLst.length; cont++)

```

```

        this.RectanguloObj(RectanguloLst[cont]);
    }

    /**
     * Dibuja un rectángulo dado el objeto rectángulo
     * @param {Rectangulo} obj - Objeto Rectangulo
     */
    RectanguloObj(obj){
        switch(obj.Tipo){
            case 1: this.Rectangulo(obj posX, obj posY, obj.Ancho, obj.Alto, obj.GrosorBorde, obj.ColorBorde);
break;
            case 2: this.RectanguloRelleno(obj posX, obj posY, obj.Ancho, obj.Alto, obj.ColorRelleno); break;
            case 3: this.RectanguloRellenoBorde(obj posX, obj posY, obj.Ancho, obj.Alto, obj.ColorRelleno,
obj.GrosorBorde, obj.ColorBorde); break;
        }
    }

    /**
     * Dibujar un polígono de 4 lados
     * @param {number} posX1 - Valor X de la primera coordenada
     * @param {number} posY1 - Valor Y de la primera coordenada
     * @param {number} posX2 - Valor X de la segunda coordenada
     * @param {number} posY2 - Valor Y de la segunda coordenada
     * @param {number} posX3 - Valor X de la tercera coordenada
     * @param {number} posY3 - Valor Y de la tercera coordenada
     * @param {number} posX4 - Valor X de la cuarta coordenada
     * @param {number} posY4 - Valor Y de la cuarta coordenada
     * @param {number} GrosorBorde - Grosor del borde del rectángulo
     * @param {string} ColorBorde - Color expresado en hexadecimal para el borde
    */
    Poligono4Lados(posX1, posY1, posX2, posY2, posX3, posY3, posX4, posY4, GrosorBorde, ColorBorde){
        this.Contexto.beginPath();
        this.Contexto.lineWidth = GrosorBorde;
        this.Contexto.strokeStyle = ColorBorde;
        this.Contexto.moveTo(posX1, posY1); //Mueve el cursor a la posición
        this.Contexto.lineTo(posX2, posY2);
        this.Contexto.lineTo(posX3, posY3);
        this.Contexto.lineTo(posX4, posY4);
        this.Contexto.lineTo(posX1, posY1);
        this.Contexto.stroke(); //Hace visible el polígono
        this.Contexto.closePath();
    }

    /**
     * Dibujar un polígono de 4 lados relleno
     * @param {number} posX1 - Valor X de la primera coordenada
     * @param {number} posY1 - Valor Y de la primera coordenada
     * @param {number} posX2 - Valor X de la segunda coordenada
     * @param {number} posY2 - Valor Y de la segunda coordenada
     * @param {number} posX3 - Valor X de la tercera coordenada
     * @param {number} posY3 - Valor Y de la tercera coordenada
     * @param {number} posX4 - Valor X de la cuarta coordenada
     * @param {number} posY4 - Valor Y de la cuarta coordenada
     * @param {string} ColorRelleno - Color expresado en hexadecimal para el relleno
    */
    Poligono4LadosRelleno(posX1, posY1, posX2, posY2, posX3, posY3, posX4, posY4, ColorRelleno){
        this.Contexto.beginPath();
        this.Contexto.fillStyle = ColorRelleno;
        this.Contexto.moveTo(posX1, posY1); //Mueve el cursor a la posición
        this.Contexto.lineTo(posX2, posY2);
        this.Contexto.lineTo(posX3, posY3);
        this.Contexto.lineTo(posX4, posY4);
        this.Contexto.lineTo(posX1, posY1);
        this.Contexto.fill(); //Rellena con el color
        this.Contexto.closePath();
    }

    /**
     * Dibujar un polígono de 4 lados relleno
     * @param {number} posX1 - Valor X de la primera coordenada
     * @param {number} posY1 - Valor Y de la primera coordenada
     * @param {number} posX2 - Valor X de la segunda coordenada
     * @param {number} posY2 - Valor Y de la segunda coordenada
     * @param {number} posX3 - Valor X de la tercera coordenada
     * @param {number} posY3 - Valor Y de la tercera coordenada
     * @param {number} posX4 - Valor X de la cuarta coordenada
     * @param {number} posY4 - Valor Y de la cuarta coordenada
     * @param {string} ColorRelleno - Color expresado en hexadecimal para el relleno
     * @param {number} GrosorBorde - Grosor del borde del rectángulo
     * @param {string} ColorBorde - Color expresado en hexadecimal para el borde
    */
    Poligono4LadosRellenoBorde(posX1, posY1, posX2, posY2, posX3, posY3, posX4, posY4, ColorRelleno, GrosorBorde,
ColorBorde){
        this.Poligono4LadosRelleno(posX1, posY1, posX2, posY2, posX3, posY3, posX4, posY4, ColorRelleno);
        this.Poligono4Lados(posX1, posY1, posX2, posY2, posX3, posY3, posX4, posY4, GrosorBorde, ColorBorde);
    }
}

```

```

/**
 * Dibuja una lista de polígonos de 4 lados
 * @param {Poligono4Lados[]} Poligono4LadosLst - Lista de objetos de la clase Poligono4Lados
 */
Poligono4LadosLista(Poligono4LadosLst){
    for(let cont = 0; cont < Poligono4LadosLst.length; cont++)
        this.Poligono4LadosObj(Poligono4LadosLst[cont]);
}

/**
 * Dibuja un polígono de 4 lados dado el objeto polígono de 4 lados
 * @param {Poligono4Lados} obj - Objeto Poligono4Lados
 */
Poligono4LadosObj(obj){
    switch(obj.Tipo){
        case 1: this.Poligono4Lados(obj.posX1, obj.posY1, obj.posX2, obj.posY2, obj.posX3, obj.posY3,
obj.posX4, obj.posY4, obj.GrosorBorde, obj.ColorBorde); break;
        case 2: this.Poligono4LadosRelleno(obj.posX1, obj.posY1, obj.posX2, obj.posY2, obj.posX3, obj.posY3,
obj.posX4, obj.posY4, obj.ColorRelleno); break;
        case 3: this.Poligono4LadosRellenoBorde(obj.posX1, obj.posY1, obj.posX2, obj.posY2, obj.posX3,
obj.posY3, obj.posX4, obj.posY4, obj.ColorRelleno, obj.GrosorBorde, obj.ColorBorde); break;
    }
}

/**
 * Dibuja un polígono dados los puntos
 * @param {Punto[]} Puntos - Lista de Puntos
 * @param {number} GrosorBorde - Grosor del borde
 * @param {string} ColorBorde - Color expresado en hexadecimal para el borde
 */
Poligono(Puntos, GrosorBorde, ColorBorde){
    this.Contexto.beginPath();
    this.Contexto.lineWidth = GrosorBorde;
    this.Contexto.strokeStyle = ColorBorde;
    this.Contexto.moveTo(Puntos[0].posX, Puntos[0].posY); //Punto inicial
    for(let cont = 1; cont < Puntos.length; cont++)
        this.Contexto.lineTo(Puntos[cont].posX, Puntos[cont].posY);
    this.Contexto.lineTo(Puntos[0].posX, Puntos[0].posY);
    this.Contexto.stroke(); //Hace visible el polígono
    this.Contexto.closePath();
}

/**
 * Dibuja un polígono relleno dados los puntos
 * @param {Punto[]} Puntos - Lista de Puntos
 * @param {string} ColorRelleno - Color expresado en hexadecimal para el relleno
 */
PoligonoRelleno(Puntos, ColorRelleno){
    this.Contexto.beginPath();
    this.Contexto.fillStyle = ColorRelleno;
    this.Contexto.moveTo(Puntos[0].posX, Puntos[0].posY); //Punto inicial
    for(let cont = 1; cont < Puntos.length; cont++)
        this.Contexto.lineTo(Puntos[cont].posX, Puntos[cont].posY);
    this.Contexto.fill(); //Rellena con el color
    this.Contexto.closePath();
}

/**
 * Dibujar polígono relleno con borde
 * @param {Punto[]} Puntos - Lista de Puntos
 * @param {string} ColorRelleno - Color expresado en hexadecimal para el relleno
 * @param {number} GrosorBorde - Grosor del borde
 * @param {string} ColorBorde - Color expresado en hexadecimal para el borde
 */
PoligonoRellenoBorde(Puntos, ColorRelleno, GrosorBorde, ColorBorde){
    this.PoligonoRelleno(Puntos, ColorRelleno);
    this.Poligono(Puntos, GrosorBorde, ColorBorde);
}

/**
 * Dibuja una curva Bézier
 * @param {number} IniX - Valor X de la coordenada inicial
 * @param {number} IniY - Valor Y de la coordenada inicial
 * @param {number} ControlX1 - Valor X de la primera coordenada de control
 * @param {number} ControlY1 - Valor Y de la primera coordenada de control
 * @param {number} ControlX2 - Valor X de la segunda coordenada de control
 * @param {number} ControlY2 - Valor Y de la segunda coordenada de control
 * @param {number} FinalX3 - Valor X de la coordenada final
 * @param {number} FinalY3 - Valor Y de la coordenada final
 * @param {number} GrosorLinea - Grosor de la línea
 * @param {string} ColorLinea - Color expresado en hexadecimal para la linea
 */
Bezier(IniX, IniY, ControlX1, ControlY1, ControlX2, ControlY2, FinalX3, FinalY3, GrosorLinea, ColorLinea){
    this.Contexto.beginPath();
    this.Contexto.lineWidth = GrosorLinea;
    this.Contexto.strokeStyle = ColorLinea;
    this.Contexto.moveTo(IniX, IniY); //Punto inicial
}

```

```

        this.Contexto.bezierCurveTo(ControlX1, ControlY1, ControlX2, ControlY2, FinalX3, FinalY3);
        this.Contexto.stroke(); //Hace visible la curva
        this.Contexto.closePath();
    }

    /**
     * Dibuja un arco
     * @param {number} posX - Valor X de la coordenada central del arco
     * @param {number} posY - Valor Y de la coordenada central del arco
     * @param {number} Radio - Radio que tendrá el arco
     * @param {number} angIni - Ángulo inicial en grados del arco
     * @param {number} angFin - Ángulo final en grados del arco
     * @param {number} GrosorLinea - Grosor de la línea
     * @param {string} ColorLinea - Color expresado en hexadecimal para la línea
     * @param {boolean} DibujaContraReloj - Si se dibuja el arco a contrareloj (true) o siguiendo las manecillas
del reloj (false)
    */
    Arco(posX, posY, Radio, angIni, angFin, GrosorLinea, ColorLinea, DibujaContraReloj){
        this.Contexto.beginPath();
        this.Contexto.lineWidth = GrosorLinea;
        this.Contexto.strokeStyle = ColorLinea;
        let AnguloIni = angIni * Math.PI / 180; //Angulo inicial (0 grados) en radianes
        let AnguloFin = angFin * Math.PI / 180; //Angulo final (360 grados) en radianes
        this.Contexto.arc(posX, posY, Radio, AnguloIni, AnguloFin, DibujaContraReloj);
        this.Contexto.stroke(); //Hace visible la curva
        this.Contexto.closePath();
    }

    /**
     * Dibuja una curva Cuadrática
     * @param {number} IniX - Valor X de la coordenada inicial de la curva
     * @param {number} IniY - Valor Y de la coordenada inicial de la curva
     * @param {number} ControlX1 - Valor X de la coordenada de control de la curva
     * @param {number} ControlY1 - Valor Y de la coordenada de control de la curva
     * @param {number} FinalX2 - Valor X de la coordenada final de la curva
     * @param {number} FinalY2 - Valor Y de la coordenada final de la curva
     * @param {number} GrosorLinea - Grosor de la línea
     * @param {string} ColorLinea - Color expresado en hexadecimal para la línea
    */
    CurvaCuadratica(IniX, IniY, ControlX1, ControlY1, FinalX2, FinalY2, GrosorLinea, ColorLinea){
        this.Contexto.beginPath();
        this.Contexto.lineWidth = GrosorLinea;
        this.Contexto.strokeStyle = ColorLinea;
        this.Contexto.moveTo(IniX, IniY); //Punto inicial
        this.Contexto.quadraticCurveTo(ControlX1, ControlY1, FinalX2, FinalY2);
        this.Contexto.stroke(); //Hace visible la curva
        this.Contexto.closePath();
    }

    /**
     * Dibuja una elipse
     * @param {number} posX - Coordenada X del centro de la elipse
     * @param {number} posY - Coordenada Y del centro de la elipse
     * @param {number} RadioX - Radio en el eje X de la elipse
     * @param {number} RadioY - Radio en el eje Y de la elipse
     * @param {number} RotaElipse - Rotación de la elipse en grados
     * @param {number} angIni - Ángulo inicial en grados de dibujo de la elipse
     * @param {number} angFin - Ángulo final en grados de dibujo de la elipse
     * @param {boolean} ContraReloj - Si se dibuja la elipse a contrareloj (true) o siguiendo las manecillas del
reloj (false)
     * @param {number} GrosorBorde - Grosor del borde
     * @param {string} ColorBorde - Color expresado en hexadecimal para el borde
    */
    Elipse(posX, posY, RadioX, RadioY, RotaElipse, angIni, angFin, ContraReloj, GrosorBorde, ColorBorde){
        this.Contexto.beginPath();
        this.Contexto.lineWidth = GrosorBorde;
        this.Contexto.strokeStyle = ColorBorde;
        let AnguloIni = angIni * Math.PI / 180; //Angulo inicial (0 grados) en radianes
        let AnguloFin = angFin * Math.PI / 180; //Angulo final (360 grados) en radianes
        let AnguloRota = RotaElipse * Math.PI / 180;
        this.Contexto.ellipse(posX, posY, RadioX, RadioY, AnguloRota, AnguloIni, AnguloFin, ContraReloj);
        this.Contexto.stroke(); //Hace visible la elipse
        this.Contexto.closePath();
    }

    /**
     * Dibuja una elipse rellena
     * @param {number} posX - Coordenada X del centro de la elipse
     * @param {number} posY - Coordenada Y del centro de la elipse
     * @param {number} RadioX - Radio en el eje X de la elipse
     * @param {number} RadioY - Radio en el eje Y de la elipse
     * @param {number} RotaElipse - Rotación de la elipse en grados
     * @param {number} angIni - Ángulo inicial en grados de dibujo de la elipse
     * @param {number} angFin - Ángulo final en grados de dibujo de la elipse
     * @param {boolean} ContraReloj - Si se dibuja la elipse a contrareloj (true) o siguiendo las manecillas del
reloj (false)
     * @param {string} ColorRelleno - Color expresado en hexadecimal para el relleno
    */

```

```

/*
ElipseRellena(posX, posY, RadioX, RadioY, RotaEllipse, angIni, angFin, ContraReloj, ColorRelleno){
    this.Contexto.beginPath();
    this.Contexto.fillStyle = ColorRelleno;
    let AnguloIni = angIni * Math.PI / 180; //Angulo inicial (0 grados) en radianes
    let AnguloFin = angFin * Math.PI / 180; //Angulo final (360 grados) en radianes
    let AnguloRota = RotaEllipse * Math.PI / 180;
    this.Contexto.ellipse(posX, posY, RadioX, RadioY, AnguloRota, AnguloIni, AnguloFin, ContraReloj);
    this.Contexto.fill(); //Rellena con el color
    this.Contexto.closePath();
}

/**
 * Dibuja una elipse rellena con borde
 * @param {number} posX - Coordenada X del centro de la elipse
 * @param {number} posY - Coordenada Y del centro de la elipse
 * @param {number} RadioX - Radio en el eje X de la elipse
 * @param {number} RadioY - Radio en el eje Y de la elipse
 * @param {number} RotaEllipse - Rotación de la elipse en grados
 * @param {number} angIni - Ángulo inicial en grados de dibujo de la elipse
 * @param {number} angFin - Ángulo final en grados de dibujo de la elipse
 * @param {boolean} ContraReloj - Si se dibuja la elipse a contrareloj (true) o siguiendo las manecillas del
reloj (false)
 * @param {string} ColorRelleno - Color expresado en hexadecimal para el relleno
 * @param {number} GrosorBorde - Grosor del borde
 * @param {string} ColorBorde - Color expresado en hexadecimal para el borde
 */
ElipseRellenaBorde(posX, posY, RadioX, RadioY, RotaEllipse, angIni, angFin, ContraReloj, ColorRelleno,
GrosorBorde, ColorBorde){
    this.Elipse(posX, posY, RadioX, RadioY, RotaEllipse, angIni, angFin, ContraReloj, GrosorBorde,
ColorBorde);
    this.ElipseRellena(posX, posY, RadioX, RadioY, RotaEllipse, angIni, angFin, ContraReloj, ColorRelleno);
}

/**
 * Dibuja una lista de elipses
 * @param {Elipse[]} ElipseLst - Lista de objetos de la clase Elipse
 */
ElipseLista(ElipseLst){
    for(let cont = 0; cont < ElipseLst.length; cont++){
        this.ElipseObj(ElipseLst[cont]);
    }
}

/**
 * Dibuja una elipse dado el objeto elipse
 * @param {Elipse} obj - Objeto Elipse
 */
ElipseObj(obj){
    switch(obj.Tipo){
        case 1: this.Elipse(obj.posX, obj.posY, obj.RadioX, obj.RadioY, obj.RotaFigura, obj.angIni,
obj.angFin, obj.Reloj, obj.GrosorBorde, obj.ColorBorde); break;
        case 2: this.ElipseRellena(obj.posX, obj.posY, obj.RadioX, obj.RadioY, obj.RotaFigura, obj.angIni,
obj.angFin, obj.Reloj, obj.ColorRelleno); break;
        case 3: this.ElipseRellenaBorde(obj.posX, obj.posY, obj.RadioX, obj.RadioY, obj.RotaFigura,
obj.angIni, obj.angFin, obj.Reloj, obj.ColorRelleno, obj.GrosorBorde, obj.ColorBorde); break;
    }
}

/**
 * Dibuja un círculo
 * @param {number} posX - Coordenada X del centro del círculo
 * @param {number} posY - Coordenada Y del centro del círculo
 * @param {number} Radio - Radio del círculo
 * @param {number} GrosorBorde - Grosor del perímetro del círculo
 * @param {string} ColorBorde - Color expresado en hexadecimal para el borde
 */
Circulo(posX, posY, Radio, GrosorBorde, ColorBorde){
    this.Contexto.beginPath();
    this.Contexto.lineWidth = GrosorBorde;
    this.Contexto.strokeStyle = ColorBorde;
    this.Contexto.arc(posX, posY, Radio, 0, 2 * Math.PI, false);
    this.Contexto.stroke(); //Dibuja el círculo
    this.Contexto.closePath();
}

/**
 * Dibujar un círculo relleno
 * @param {number} posX - Coordenada X del centro del círculo
 * @param {number} posY - Coordenada Y del centro del círculo
 * @param {number} Radio - Radio del círculo
 * @param {string} ColorRelleno - Color expresado en hexadecimal para el relleno
 */
CirculoRelleno(posX, posY, Radio, ColorRelleno){
    this.Contexto.beginPath();
    this.Contexto.fillStyle = ColorRelleno;
    this.Contexto.arc(posX, posY, Radio, 0, 2 * Math.PI, false);
    this.Contexto.fill(); //Rellena con el color
}

```

```

        this.Contexto.closePath();

}

/***
 * Dibujar un círculo relleno con borde
 * @param {number} posX - Coordenada X del centro del círculo
 * @param {number} posY - Coordenada Y del centro del círculo
 * @param {number} Radio - Radio del círculo
 * @param {string} ColorRelleno - Color expresado en hexadecimal para el relleno
 * @param {number} GrosorBorde - Grosor del borde
 * @param {string} ColorBorde - Color expresado en hexadecimal para el borde
 */
CirculoRellenoBorde(posX, posY, Radio, ColorRelleno, GrosorBorde, ColorBorde) {
    this.CirculoRelleno(posX, posY, Radio, ColorRelleno);
    this.Circulo(posX, posY, Radio, GrosorBorde, ColorBorde);
}

/***
 * Dibuja una lista de círculos
 * @param {Circulo[]} CirculoLst - Lista de círculos
 */
CirculoLista(CirculoLst){
    for(let cont = 0; cont < CirculoLst.length; cont++)
        this.CirculoObj(CirculoLst[cont]);
}

/***
 * Dibuja un círculo dado el objeto círculo
 * @param {Circulo} obj - Objeto círculo
 */
CirculoObj(obj){
    switch(obj.Tipo){
        case 1: this.Circulo(obj.posX, obj.posY, obj.Radio, obj.GrosorBorde, obj.ColorBorde); break;
        case 2: this.CirculoRelleno(obj.posX, obj.posY, obj.Radio, obj.ColorRelleno); break;
        case 3: this.CirculoRellenoBorde(obj.posX, obj.posY, obj.Radio, obj.ColorRelleno, obj.GrosorBorde,
obj.ColorBorde); break;
    }
}

/***
 * Dibuja una cadena de texto con un color de borde
 * @param {number} posX - Posición en X del texto
 * @param {number} posY - Posición en Y del texto
 * @param {string} Texto - Texto a mostrar en pantalla
 * @param {string} Color - Color expresado en hexadecimal para el texto
 * @param {string} Fuente - Características de la fuente a usar
 */
DibujarCadenas(posX, posY, Texto, Color, Fuente){
    this.Contexto.beginPath();
    this.Contexto.font = Fuente; //Tipo, tamaño, fuente
    this.Contexto.strokeStyle = Color;
    this.Contexto.strokeText(Texto, posX, posY); //Inferior
    this.Contexto.closePath();
}

/***
 * Dibuja una cadena de texto con color de relleno
 * @param {number} posX - Posición en X del texto
 * @param {number} posY - Posición en Y del texto
 * @param {string} Texto - Texto a mostrar en pantalla
 * @param {string} Color - Color expresado en hexadecimal para el relleno del texto
 * @param {string} Fuente - Características de la fuente a usar
 */
DibujarCadenasRelleno(posX, posY, Texto, Color, Fuente){
    this.Contexto.beginPath();
    this.Contexto.font = Fuente; //Tipo, tamaño, fuente
    this.Contexto.fillStyle = Color;
    this.Contexto.fillText(Texto, posX, posY); //Superior
    this.Contexto.closePath();
}

// ===== GRAFICO MATEMATICO 2D =====

/***
 * Hace el gráfico matemático generado por la ecuación Y=F(X)
 * @param {string} Ecuacion - Expresión matemática del tipo Y = F(X), por ejemplo Y = sen(X)-cos(2*X)+tan(X^3)
. Se pueden usar las siguientes funciones sen() seno, cos() coseno, tan() tangente, abs() valor absoluto, asn()
arcoseno, acs() arcocoseno, atn() arcotangente, log() logaritmo natural, cei() valor techo, exp() exponencial, sqr()
raíz cuadrada; + suma, - resta, * multiplicación, / división, ^ potencia; paréntesis y una variable independiente X
 * @param {number} Xini - Valor X inicial desde donde se evalúa la ecuación
 * @param {number} Xfin - Valor X final hasta donde se evalúa la ecuación
 * @param {number} numPuntos - Total de puntos que se van a calcular entre Xini y Xfin
 * @param {number} Grosor - Grosor de la línea
 * @param {string} Color - Color expresado en hexadecimal para la línea
 */
Matematico2D(Ecuacion, Xini, Xfin, numPuntos, Grosor, Color){
    //Llama al evaluador de expresiones
}

```

```

let evaluador2022 = new Evaluador4();
evaluador2022.Analizar(Ecuacion);

    //Calcula los puntos de la ecuación a graficar
let pasoX = (Xfin - Xini) / numPuntos;
let Ymin = Number.MAX_VALUE; //El mínimo valor de Y obtenido
let Ymax = -Ymin; //El máximo valor de Y obtenido
let maximoXreal = -Ymin; //El máximo valor de X (difiere de Xfin)

let puntos = [];
for (let X = Xini; X <= Xfin; X += pasoX) {
    evaluador2022.DarValorVariable('x', X);
    let valY = -1*evaluador2022.Evaluar(); //Se invierte el valor porque el eje Y aumenta hacia abajo
    if (valY > Ymax) Ymax = valY;
    if (valY < Ymin) Ymin = valY;
    if (X > maximoXreal) maximoXreal = X;
    puntos.push(new Punto(X, valY));
}

//;OJO! X puede que no llegue a ser Xfin, por lo que la variable maximoXreal almacena el valor máximo de
X

//Calcula los puntos a poner en la pantalla
let convierteX = this.LienzoAncho() / (maximoXreal - Xini);
let convierteY = this.LienzoAlto() / (Ymax - Ymin);

for (let cont = 0; cont < puntos.length; cont++) {
    puntos[cont].posX = Math.round(convierteX * (puntos[cont].posX - Xini));
    puntos[cont].posY = Math.round(convierteY * (puntos[cont].posY - Ymin));
}

//Hace el gráfico
this.LineasPuntos(puntos, Grosor, Color);
}

/**
 * Convierte los números RGB (Red, Green, Blue) en su equivalente hexadecimal de color
 * @param {number} Rojo - Valor de 0 a 254 de rojo
 * @param {number} Verde - Valor de 0 a 254 de verde
 * @param {number} Azul - Valor de 0 a 254 de azul
 * @return {string} Color en hexadecimal
 */
ColorRGB(Rojo, Verde, Azul){
    return "#" + this.Hexadecimal(Rojo) + this.Hexadecimal(Verde) + this.Hexadecimal(Azul);
}

/**
 * Convierte un número en hexadecimal
 * @param {number} Numero - Número a convertir
 * @returns {string} - Número en hexadecimal
 */
Hexadecimal(Numero){
    if (isNaN(Numero)) return "00";
    return "0123456789ABCDEF".charAt((Numero-Numero%16)/16)+"0123456789ABCDEF".charAt(Numero%16);
}

/**
 * Retorna un color al azar en formato hexadecimal
 * @returns {string} - Color en formato hexadecimal
 */
ColorAzar(){
    let Rojo = Math.round(Math.random() * 255);
    let Verde = Math.round(Math.random() * 255);
    let Azul = Math.round(Math.random() * 255);
    return this.ColorRGB(Rojo, Verde, Azul);
}

/** Clase para mantener un conjunto de puntos
 * @Author Rafael Alberto Moreno Parra
 * @Version 1.0 (05 de enero de 2024)
 * */
class Punto {
    /**
     * Constructor
     * @param {number} posX - Valor X de la coordenada
     * @param {number} posY - Valor Y de la coordenada
     */
    constructor(posX, posY){
        this.posX = posX;
        this.posY = posY;
    }
}

/** Clase para mantener un conjunto de líneas
 * @Author Rafael Alberto Moreno Parra

```

```

* @Version 1.0 (05 de enero de 2024)
* */
class Linea {
    /**
     * Constructor
     * @param {number} posXini - Valor X de la coordenada de inicio
     * @param {number} posYini - Valor Y de la coordenada de inicio
     * @param {number} posXfin - Valor X de la coordenada final
     * @param {number} posYfin - Valor Y de la coordenada final
     * @param {number} Grosor - Grosor de la línea
     * @param {string} Color - Color expresado en hexadecimal de la línea
    */
    constructor(posXini, posYini, posXfin, posYfin, Grosor, Color) {
        this.posXini = posXini;
        this.posYini = posYini;
        this.posXfin = posXfin;
        this.posYfin = posYfin;
        this.Grosor = Grosor;
        this.Color = Color;
    }

    /**
     * Trasladar una línea
     * @param {number} mueveX - Traslado en el eje X
     * @param {number} mueveY - Traslado en el eje Y
    */
    Trasladar(mueveX, mueveY) {
        this.posXini += mueveX;
        this.posYini += mueveY;
        this.posXfin += mueveX;
        this.posYfin += mueveY;
    }

    /**
     * Girar la línea calculando su centroide
     * @param {number} angulo - Ángulo en grados de giro
    */
    Girar(angulo) {
        let anguloR = angulo * Math.PI / 180;

        //Giro de las dos coordenadas
        let posXinig = Math.round(this.posXini * Math.cos(anguloR) - this.posYini * Math.sin(anguloR));
        let posYinig = Math.round(this.posXini * Math.sin(anguloR) + this.posYini * Math.cos(anguloR));
        let posXfing = Math.round(this.posXfin * Math.cos(anguloR) - this.posYfin * Math.sin(anguloR));
        let posYfing = Math.round(this.posXfin * Math.sin(anguloR) + this.posYfin * Math.cos(anguloR));

        //Giro del centroide
        let centroX = (this.posXini + this.posXfin) / 2;
        let centroY = (this.posYini + this.posYfin) / 2;
        let centroXg = Math.round(centroX * Math.cos(anguloR) - centroY * Math.sin(anguloR));
        let centroYg = Math.round(centroX * Math.sin(anguloR) + centroY * Math.cos(anguloR));

        //¿Cuánto se desplazó el centroide?
        let desplazaX = centroXg - centroX;
        let desplazaY = centroYg - centroY;

        //Retira el desplazamiento
        this.posXini = posXinig - desplazaX;
        this.posYini = posYinig - desplazaY;
        this.posXfin = posXfing - desplazaX;
        this.posYfin = posYfing - desplazaY;
    }
}

/** Clase para mantener un conjunto de triángulos
 * @Author Rafael Alberto Moreno Parra
 * @Version 1.0 (05 de enero de 2024)
 * */
class Triangulo {
    /**
     * Constructor
     * @param {number} posX1 - Valor X de la primera coordenada
     * @param {number} posY1 - Valor Y de la primera coordenada
     * @param {number} posX2 - Valor X de la segunda coordenada
     * @param {number} posY2 - Valor Y de la segunda coordenada
     * @param {number} posX3 - Valor X de la tercera coordenada
     * @param {number} posY3 - Valor Y de la tercera coordenada
     * @param {number} Tipo - Tipo triángulo: 1. Sólo perímetro, 2. Relleno, 3. Perímetro de un color y relleno de otro color
     * @param {number} GrosorBorde - Grosor del borde
     * @param {string} ColorBorde - Color expresado en hexadecimal del borde
     * @param {string} ColorRelleno - Color expresado en hexadecimal del relleno
    */
    constructor(posX1, posY1, posX2, posY2, posX3, posY3, Tipo, GrosorBorde, ColorBorde, ColorRelleno) {
        this.posX1 = posX1;
        this.posY1 = posY1;
        this.posX2 = posX2;

```

```

        this.posY2 = posY2;
        this.posX3 = posX3;
        this.posY3 = posY3;
        this.Tipo = Tipo; //1. Solo borde, 2. Sólo relleno, 3. Borde y relleno
        this.GrosorBorde = GrosorBorde;
        this.ColorBorde = ColorBorde;
        this.ColorRelleno = ColorRelleno;
    }

    /**
     * Trasladar un triángulo
     * @param {number} mueveX - Traslado en el eje X
     * @param {number} mueveY - Traslado en el eje Y
     */
    Trasladar(mueveX, mueveY) {
        this.posX1 += mueveX;
        this.posY1 += mueveY;
        this.posX2 += mueveX;
        this.posY2 += mueveY;
        this.posX3 += mueveX;
        this.posY3 += mueveY;
    }

    /**
     * Girar el triángulo calculando su centroide
     * @param {number} angulo - Ángulo en grados de giro
     */
    Girar(angulo) {
        let anguloR = angulo * Math.PI / 180;

        //Giro de las tres coordenadas
        let posX1g = Math.round(this.posX1 * Math.cos(anguloR) - this.posY1 * Math.sin(anguloR));
        let posY1g = Math.round(this.posX1 * Math.sin(anguloR) + this.posY1 * Math.cos(anguloR));
        let posX2g = Math.round(this.posX2 * Math.cos(anguloR) - this.posY2 * Math.sin(anguloR));
        let posY2g = Math.round(this.posX2 * Math.sin(anguloR) + this.posY2 * Math.cos(anguloR));
        let posX3g = Math.round(this.posX3 * Math.cos(anguloR) - this.posY3 * Math.sin(anguloR));
        let posY3g = Math.round(this.posX3 * Math.sin(anguloR) + this.posY3 * Math.cos(anguloR));

        //Giro del centroide
        let centroX = (this.posX1 + this.posX2 + this.posX3) / 3;
        let centroY = (this.posY1 + this.posY2 + this.posY3) / 3;
        let centroXg = Math.round(centroX * Math.cos(anguloR) - centroY * Math.sin(anguloR));
        let centroYg = Math.round(centroX * Math.sin(anguloR) + centroY * Math.cos(anguloR));

        //¿Cuánto se desplazó el centroide?
        let desplazaX = centroXg - centroX;
        let desplazaY = centroYg - centroY;

        //Retira el desplazamiento
        this.posX1 = posX1g - desplazaX;
        this.posY1 = posY1g - desplazaY;
        this.posX2 = posX2g - desplazaX;
        this.posY2 = posY2g - desplazaY;
        this.posX3 = posX3g - desplazaX;
        this.posY3 = posY3g - desplazaY;
    }
}

/** Clase para mantener un conjunto de rectángulos
 * @Author Rafael Alberto Moreno Parra
 * @Version 1.0 (05 de enero de 2024)
 */
class Rectangulo {
    /**
     * Constructor
     * @param {number} posX - Valor X de la coordenada superior izquierda
     * @param {number} posY - Valor Y de la coordenada superior izquierda
     * @param {number} Ancho - Ancho del rectángulo
     * @param {number} Alto - Alto del rectángulo
     * @param {number} Tipo - Tipo rectángulo: 1. Sólo perímetro, 2. Relleno, 3. Perímetro de un color y relleno de otro color
     * @param {number} GrosorBorde - Grosor línea del borde
     * @param {string} ColorBorde - Color expresado en hexadecimal del borde
     * @param {string} ColorRelleno - Color expresado en hexadecimal del relleno
     */
    constructor(posX, posY, Ancho, Alto, Tipo, GrosorBorde, ColorBorde, ColorRelleno) {
        this.posX = posX;
        this.posY = posY;
        this.Ancho = Ancho;
        this.Alto = Alto;
        this.Tipo = Tipo; //1. Solo borde, 2. Sólo relleno, 3. Borde y relleno
        this.GrosorBorde = GrosorBorde;
        this.ColorBorde = ColorBorde;
        this.ColorRelleno = ColorRelleno;
    }
}

```

```

    * Trasladar un rectángulo
    * @param {number} mueveX - Traslado en el eje X
    * @param {number} mueveY - Traslado en el eje Y
    */
    Trasladar(mueveX, mueveY){
        this.posX += mueveX;
        this.posY += mueveY;
    }
}

/** Clase para mantener un conjunto de polígonos de 4 lados
 * @Author Rafael Alberto Moreno Parra
 * @Version 1.0 (05 de enero de 2024)
 */
class Poligono4Lados {
    /**
     * Constructor
     * @param {number} posX1 - Valor X de la primera coordenada
     * @param {number} posY1 - Valor Y de la primera coordenada
     * @param {number} posX2 - Valor X de la segunda coordenada
     * @param {number} posY2 - Valor Y de la segunda coordenada
     * @param {number} posX3 - Valor X de la tercera coordenada
     * @param {number} posY3 - Valor Y de la tercera coordenada
     * @param {number} posX4 - Valor X de la cuarta coordenada
     * @param {number} posY4 - Valor Y de la cuarta coordenada
     * @param {number} Tipo - Tipo polígono de 4 lados: 1. Sólo perímetro, 2. Relleno, 3. Perímetro de un color y
     * relleno de otro color
     * @param {number} GrosorBorde - Grosor línea del borde
     * @param {string} ColorBorde - Color expresado en hexadecimal del borde
     * @param {string} ColorRelleno - Color expresado en hexadecimal del relleno
    */
    constructor(posX1, posY1, posX2, posY2, posX3, posY3, posX4, posY4, Tipo, GrosorBorde, ColorBorde,
    ColorRelleno){
        this.posX1 = posX1;
        this.posY1 = posY1;
        this.posX2 = posX2;
        this.posY2 = posY2;
        this.posX3 = posX3;
        this.posY3 = posY3;
        this.posX4 = posX4;
        this.posY4 = posY4;
        this.Tipo = Tipo; //1. Solo borde, 2. Sólo relleno, 3. Borde y relleno
        this.GrosorBorde = GrosorBorde;
        this.ColorBorde = ColorBorde;
        this.ColorRelleno = ColorRelleno;
    }

    /**
     * Trasladar un polígono de 4 lados
     * @param {number} mueveX - Traslado en el eje X
     * @param {number} mueveY - Traslado en el eje Y
    */
    Trasladar(mueveX, mueveY){
        this.posX1 += mueveX;
        this.posY1 += mueveY;
        this.posX2 += mueveX;
        this.posY2 += mueveY;
        this.posX3 += mueveX;
        this.posY3 += mueveY;
        this.posX4 += mueveX;
        this.posY4 += mueveY;
    }

    /**
     * Girar el polígono de 4 lados calculando su centroide
     * @param {number} angulo - Ángulo en grados de giro
    */
    Girar(angulo){
        let anguloR = angulo * Math.PI / 180;

        //Giro de las cuatro coordenadas
        let posX1g = Math.round(this.posX1 * Math.cos(anguloR) - this.posY1 * Math.sin(anguloR));
        let posY1g = Math.round(this.posX1 * Math.sin(anguloR) + this.posY1 * Math.cos(anguloR));
        let posX2g = Math.round(this.posX2 * Math.cos(anguloR) - this.posY2 * Math.sin(anguloR));
        let posY2g = Math.round(this.posX2 * Math.sin(anguloR) + this.posY2 * Math.cos(anguloR));
        let posX3g = Math.round(this.posX3 * Math.cos(anguloR) - this.posY3 * Math.sin(anguloR));
        let posY3g = Math.round(this.posX3 * Math.sin(anguloR) + this.posY3 * Math.cos(anguloR));
        let posX4g = Math.round(this.posX4 * Math.cos(anguloR) - this.posY4 * Math.sin(anguloR));
        let posY4g = Math.round(this.posX4 * Math.sin(anguloR) + this.posY4 * Math.cos(anguloR));

        //Giro del centroide
        let centroX = (this.posX1 + this.posX2 + this.posX3 + this.posX4) / 4;
        let centroY = (this.posY1 + this.posY2 + this.posY3 + this.posY4) / 4;
        let centroXg = Math.round(centroX * Math.cos(anguloR) - centroY * Math.sin(anguloR));
        let centroYg = Math.round(centroX * Math.sin(anguloR) + centroY * Math.cos(anguloR));

        //¿Cuánto se desplazó el centroide?
    }
}

```

```

let desplazaX = centroXg - centroX;
let desplazaY = centroYg - centroY;

    //Retira el desplazamiento
    this.posX1 = posX1g - desplazaX;
    this.posY1 = posY1g - desplazaY;
    this.posX2 = posX2g - desplazaX;
    this.posY2 = posY2g - desplazaY;
    this.posX3 = posX3g - desplazaX;
    this.posY3 = posY3g - desplazaY;
    this.posX4 = posX4g - desplazaX;
    this.posY4 = posY4g - desplazaY;
}

/** Clase para mantener un conjunto de elipses
 * @Author Rafael Alberto Moreno Parra
 * @Version 1.0 (05 de enero de 2024)
 */
class Elipse {
    /**
     * Constructor
     * @param {number} posX - Posición X de la coordenada del centro de la elipse
     * @param {number} posY - Posición Y de la coordenada del centro de la elipse
     * @param {number} RadioX - Longitud del radio en el eje X
     * @param {number} RadioY - Longitud del radio en el eje Y
     * @param {number} RotaFigura - Giro de la elipse
     * @param {number} angIni - Ángulo inicial de dibujado de la elipse
     * @param {number} angFin - Ángulo final de dibujado de la elipse
     * @param {boolean} ContraReloj - Si el dibujado es contrareloj es true
     * @param {number} Tipo - Tipo elipse: 1. Sólo el borde, 2. Relleno, 3. Borde de un color y relleno de otro
color
     * @param {number} GrosorBorde - Grosor del borde
     * @param {string} ColorBorde - Color expresado en hexadecimal del borde
     * @param {string} ColorRelleno - Color expresado en hexadecimal del relleno
    */
    constructor(posX, posY, RadioX, RadioY, RotaFigura, angIni, angFin, ContraReloj, Tipo, GrosorBorde, ColorBorde, ColorRelleno) {
        this.posX = posX;
        this.posY = posY;
        this.RadioX = RadioX;
        this.RadioY = RadioY;
        this.RotaFigura = RotaFigura;
        this.angIni = angIni;
        this.angFin = angFin;
        this.Reloj = ContraReloj;
        this.Tipo = Tipo; //1. Solo borde, 2. Sólo relleno, 3. Borde y relleno
        this.GrosorBorde = GrosorBorde;
        this.ColorBorde = ColorBorde;
        this.ColorRelleno = ColorRelleno;
    }

    /**
     * Trasladar una elipse
     * @param {number} mueveX - Traslado en el eje X
     * @param {number} mueveY - Traslado en el eje Y
    */
    Trasladar(mueveX, mueveY) {
        this.posX += mueveX;
        this.posY += mueveY;
    }
}

/** Clase para mantener un conjunto de círculos
 * @Author Rafael Alberto Moreno Parra
 * @Version 1.0 (05 de enero de 2024)
 */
class Circulo {
    /**
     * Constructor
     * @param {number} posX - Posición X de la coordenada del centro del círculo
     * @param {number} posY - Posición Y de la coordenada del centro del círculo
     * @param {number} Radio - Radio del círculo
     * @param {number} Tipo - Tipo de círculo: 1. Sólo color en el perímetro, 2. Relleno, 3. Relleno de un color y
perímetro de otro color
     * @param {number} GrosorBorde - Grosor del perímetro
     * @param {string} ColorBorde - Color expresado en hexadecimal del borde
     * @param {string} ColorRelleno - Color expresado en hexadecimal del relleno
    */
    constructor(posX, posY, Radio, Tipo, GrosorBorde, ColorBorde, ColorRelleno) {
        this.posX = posX;
        this.posY = posY;
        this.Radio = Radio;
        this.Tipo = Tipo; //1. Solo borde, 2. Sólo relleno, 3. Borde y relleno
        this.GrosorBorde = GrosorBorde;
        this.ColorBorde = ColorBorde;
        this.ColorRelleno = ColorRelleno;
    }
}

```

```

}

/**
 * Trasladar un círculo
 * @param {number} mueveX - Traslado en el eje X
 * @param {number} mueveY - Traslado en el eje Y
 */
Trasladar(mueveX, mueveY) {
    this posX += mueveX;
    this posY += mueveY;
}
}

```

Y esta es la forma de usar esa librería:

291.html

```

<!DOCTYPE HTML><html lang="es">
<head><title>Gráficos</title></head>
<body>
<canvas id="milienzo" width="1200" height="800"></canvas>
<script src="LibreriaGrafica.js"></script>
<script>
    //Objeto para hacer los gráficos
    let Graficos = new Grafica(document.getElementById('milienzo'));

    let Color = Graficos.ColorAzar(); //Algún color al azar
    let GrosorLinea = Azar.EnteroEntre(2, 10); //Grueso entre 1 y 10

    //Trabaja con una lista de líneas
    let Lineas = [];
    Lineas.push(new Linea(150, 150, 300, 300, GrosorLinea, Color)); //Xini, Yini, Xfin, Yfin, Grueso, Color
    Lineas.push(new Linea(200, 50, 120, 350, GrosorLinea, Color));
    Lineas.push(new Linea(300, 140, 210, 290, GrosorLinea, Color));
    Lineas.push(new Linea(320, 240, 110, 220, GrosorLinea, Color));
    Lineas.push(new Linea(270, 340, 310, 260, GrosorLinea, Color));

    //Dibuja las líneas: Lista
    Graficos.LineasListas(Lineas);
</script>
</body>
</html>

```

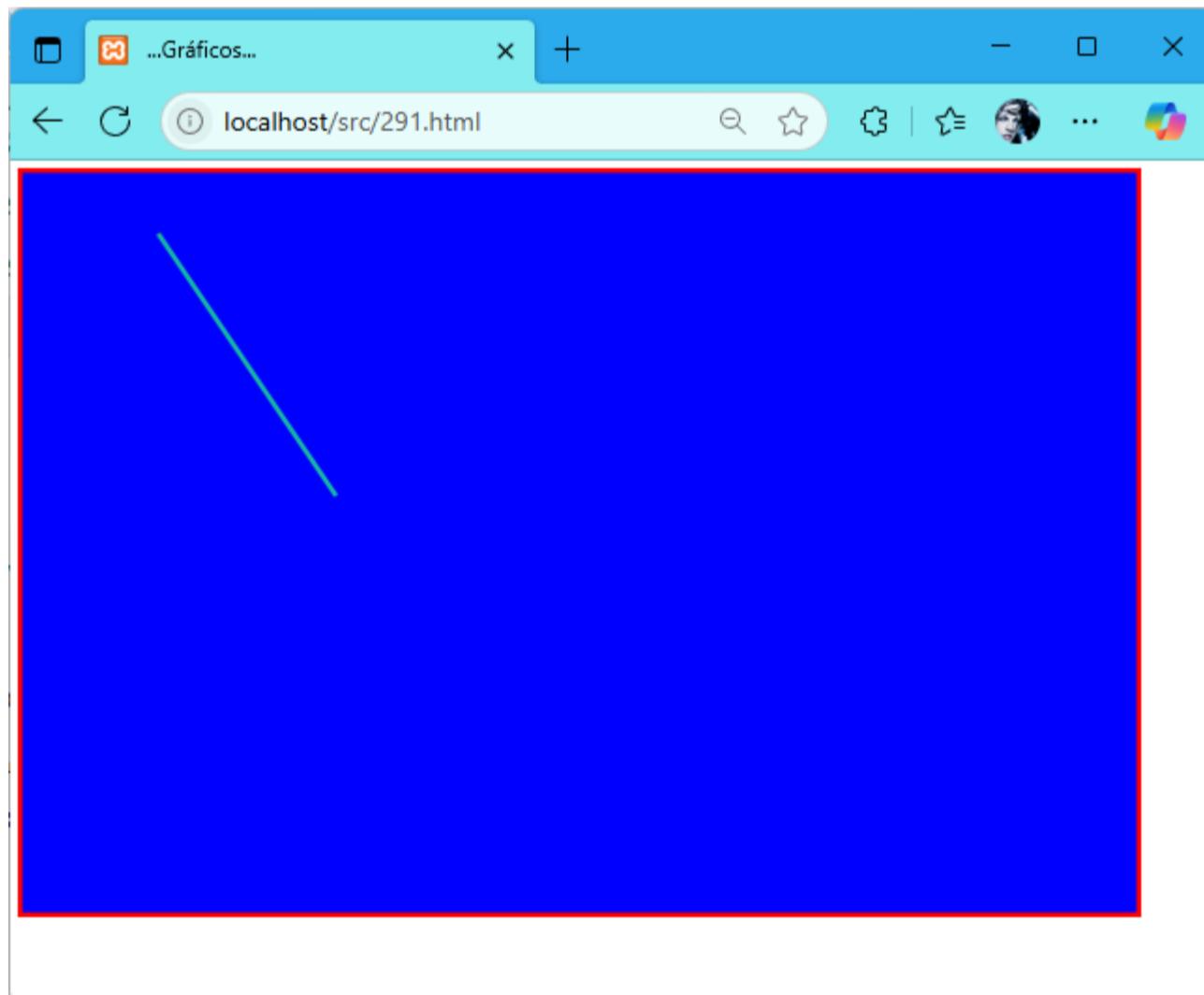


Ilustración 97: Uso de librería gráfica

```
<!DOCTYPE HTML><html lang="es">
<head><title>Gráficos</title></head>
<body>
<canvas id="milienzo" width="1200" height="800"></canvas>
<script src="LibreriaGrafica.js"></script>
<script>
    //Objeto para hacer los gráficos
    let Graficos = new Grafica(document.getElementById('milienzo'));

    //Trabaja con una lista de puntos
    let Puntos = [];
    Puntos.push(new Punto(150, 150));
    Puntos.push(new Punto(200, 50));
    Puntos.push(new Punto(300, 140));
    Puntos.push(new Punto(400, 200));
    Puntos.push(new Punto(450, 300));
    Puntos.push(new Punto(350, 350));
    Puntos.push(new Punto(300, 150));

    let Color = Graficos.ColorRGB(254, 0, 0); //Color en formato RGB
    let GrosorLinea = 2;

    //Dibuja las líneas
    Graficos.LineasPuntos(Puntos, GrosorLinea, Color);
</script>
</body>
</html>
```

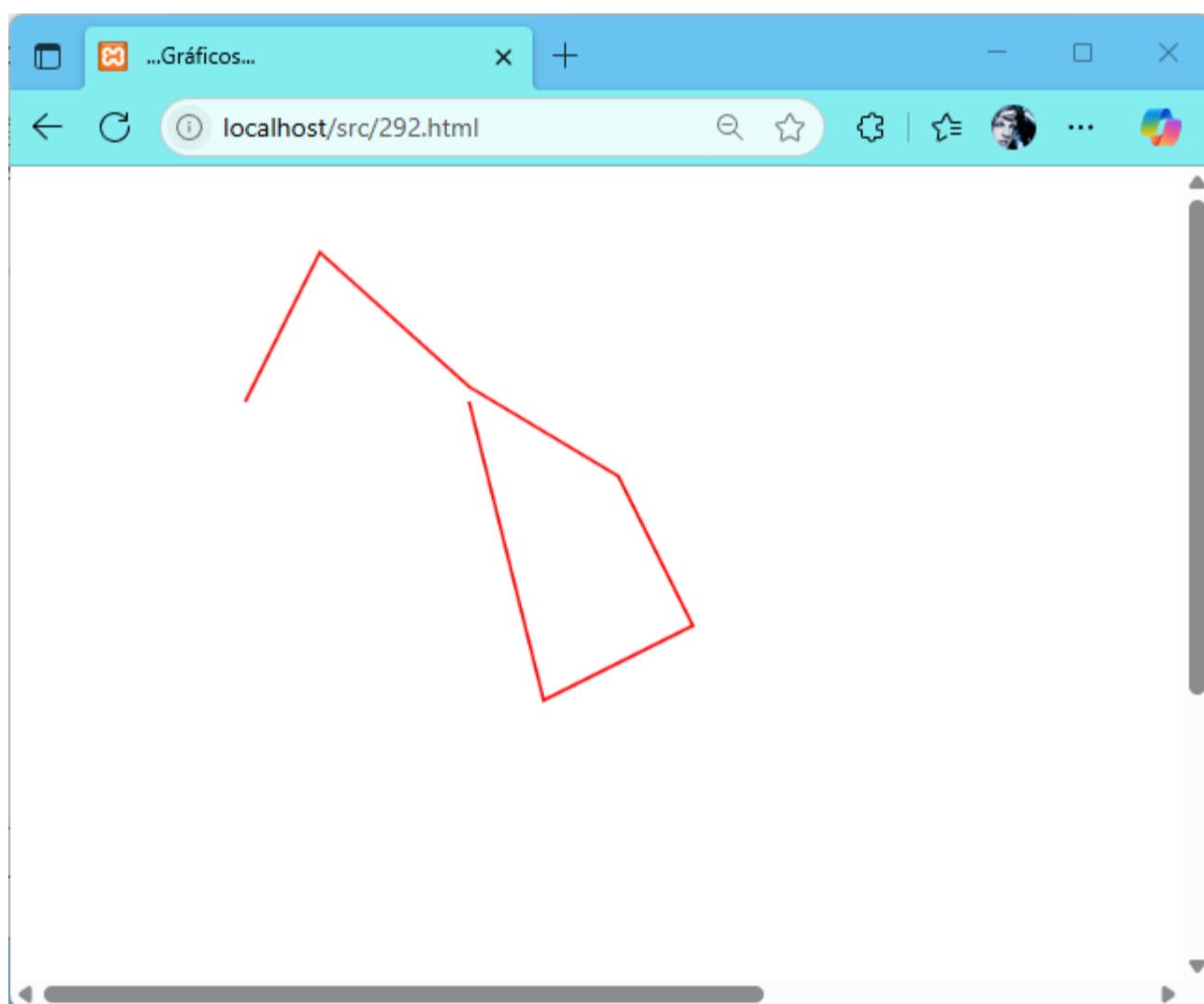


Ilustración 98: Uso de librería gráfica

```
<!DOCTYPE HTML><html lang="es">
<head><title>Gráficos</title></head>
<body>
<canvas id="milienzo" width="1200" height="800"></canvas>
<script src="LibreriaGrafica.js"></script>
<script src="AzarLibreria.js"></script>
<script>
    //Objeto para hacer los gráficos
    let Graficos = new Grafica(document.getElementById('milienzo'));

    let Color = Graficos.ColorAzar(); //Algún color al azar
    let GrosorLinea = Azar.EntreEntre(2, 10); //Grueso entre 1 y 10

    //Trabaja con una lista de líneas
    let Lineas = [];
    Lineas.push(new Linea(150, 150, 300, 300, GrosorLinea, Color)); //Xini, Yini, Xfin, Yfin, Grueso, Color
    Lineas.push(new Linea(200, 50, 120, 350, GrosorLinea, Color));
    Lineas.push(new Linea(300, 140, 210, 290, GrosorLinea, Color));
    Lineas.push(new Linea(320, 240, 110, 220, GrosorLinea, Color));
    Lineas.push(new Linea(270, 340, 310, 260, GrosorLinea, Color));

    //Dibuja las líneas: Lista
    Graficos.LineasListas(Lineas);
</script>
</body>
</html>
```

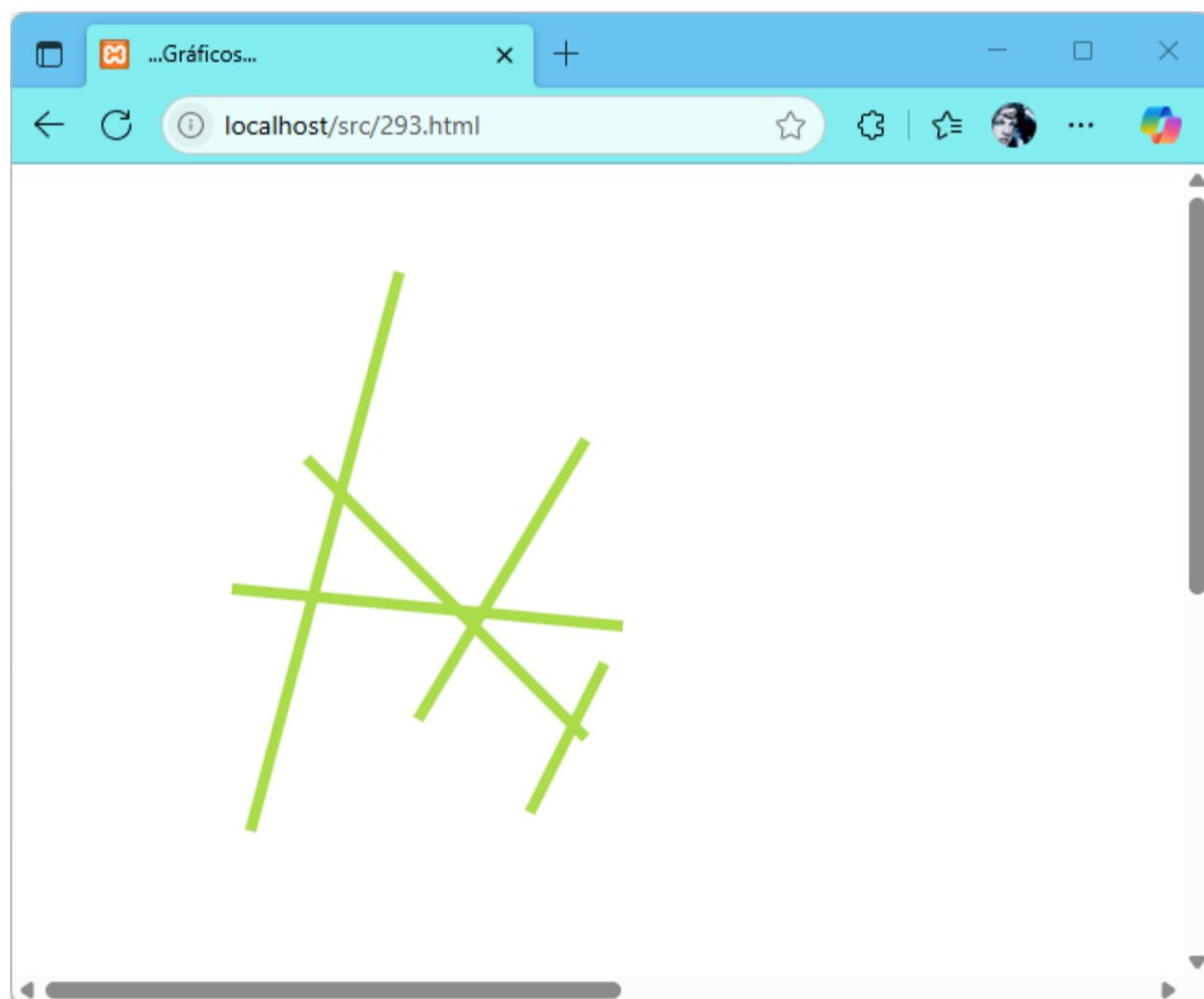


Ilustración 99: Uso de librería gráfica

```
<!DOCTYPE HTML><html lang="es">
<head><title>Gráficos</title></head>
<body>
<canvas id="milienzo" width="1200" height="800"></canvas>
<script src="LibreriaGrafica.js"></script>
<script src="AzarLibreria.js"></script>
<script>
    //Objeto para hacer los gráficos
    let Graficos = new Grafica(document.getElementById('milienzo'));

    //Lista de líneas
    let Lineas = [];
    for (let cont = 1; cont <= 30; cont++) {
        let Color = Graficos.ColorAzar(); //Algún color al azar
        let Grueso = Azar.EntreEntre(1, 10); //Grueso entre 1 y 10

        //Trabaja con una lista de líneas
        let Xini = Azar.EntreEntre(1, 800); let Yini = Azar.EntreEntre(1, 800);
        let Xfin = Azar.EntreEntre(1, 800); let Yfin = Azar.EntreEntre(1, 800);
        Lineas.push(new Linea(Xini, Yini, Xfin, Yfin, Grueso, Color));
    }

    //Dibuja las líneas: Lista
    Graficos.LineasListas(Lineas);
</script>
</body>
</html>
```

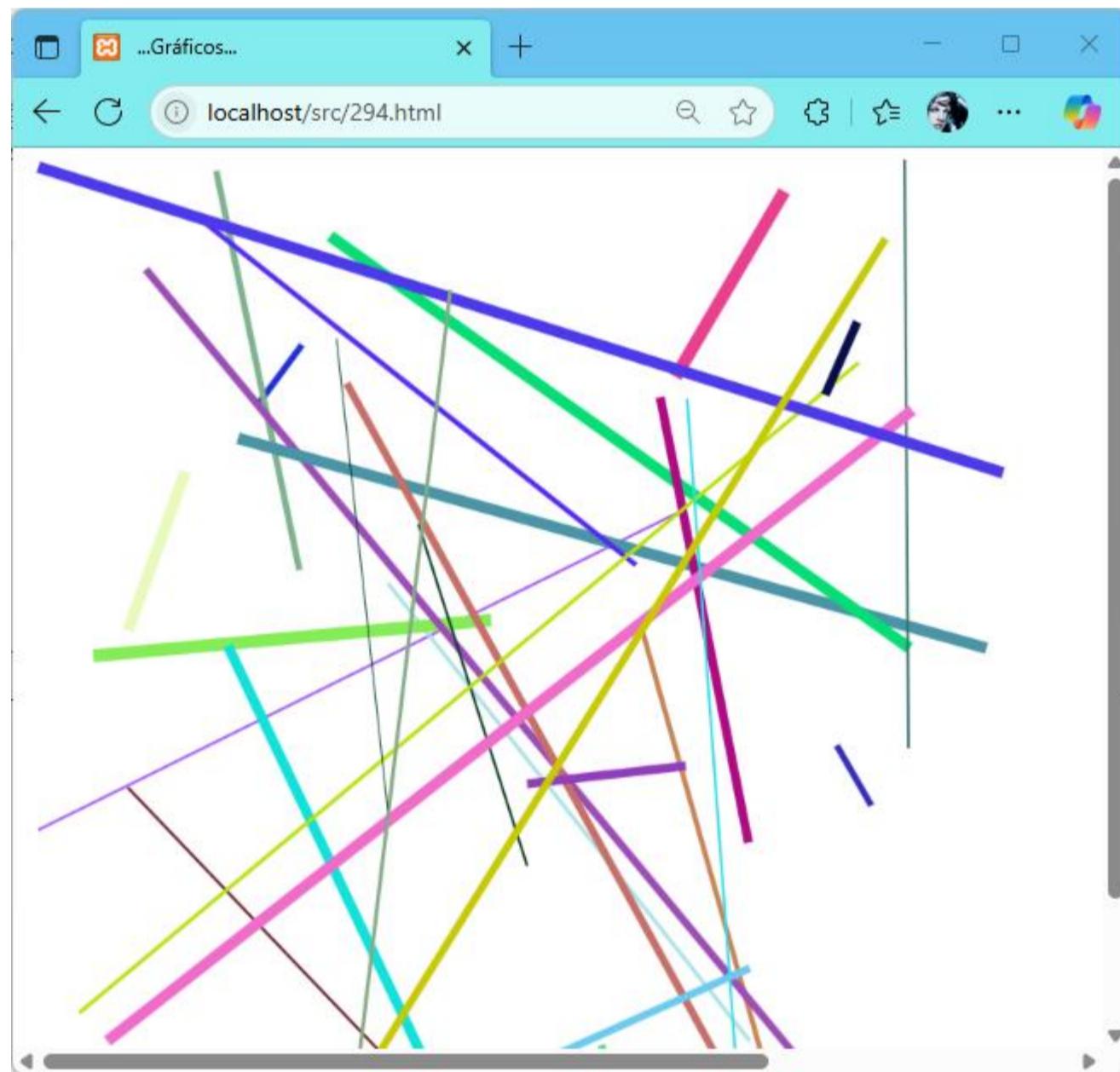


Ilustración 100: Uso de librería gráfica

```
<!DOCTYPE HTML><html lang="es">
<head><title>Gráficos</title></head>
<body>
<canvas id="milienzo" width="1200" height="800"></canvas>
<script src="LibreriaGrafica.js"></script>
<script>
    //Objeto para hacer los gráficos
    let Graficos = new Grafica(document.getElementById('milienzo'));

    //Dibuja un triángulo: posX1, posY1, posX2, posY2, posX3, posY3, Grosor, Color
    let ColorBorde = 'blue';
    let posX1 = 10, posY1 = 10;
    let posX2 = 90, posY2 = 130;
    let posX3 = 220, posY3 = 30;
    let Grosor = 4;
    Graficos.Triangulo(posX1, posY1, posX2, posY2, posX3, posY3, Grosor, ColorBorde);

    //Dibuja un triángulo relleno: posX1, posY1, posX2, posY2, posX3, posY3, ColorRelleno
    let ColorRelleno = 'gray';
    posX1 = 10; posY1 = 210;
    posX2 = 190; posY2 = 330;
    posX3 = 420; posY3 = 130;
    Graficos.TrianguloRelleno(posX1, posY1, posX2, posY2, posX3, posY3, ColorRelleno);

    //Dibuja un triángulo relleno con borde: posX1, posY1, posX2, posY2, posX3, posY3, ColorRelleno, GrosorBorde, ColorBorde
    posX1 = 10; posY1 = 370;
    posX2 = 190; posY2 = 460;
    posX3 = 420; posY3 = 380;
    Graficos.TrianguloRellenoBorde(posX1, posY1, posX2, posY2, posX3, posY3, ColorRelleno, Grosor, ColorBorde);
</script>
</body>
</html>
```

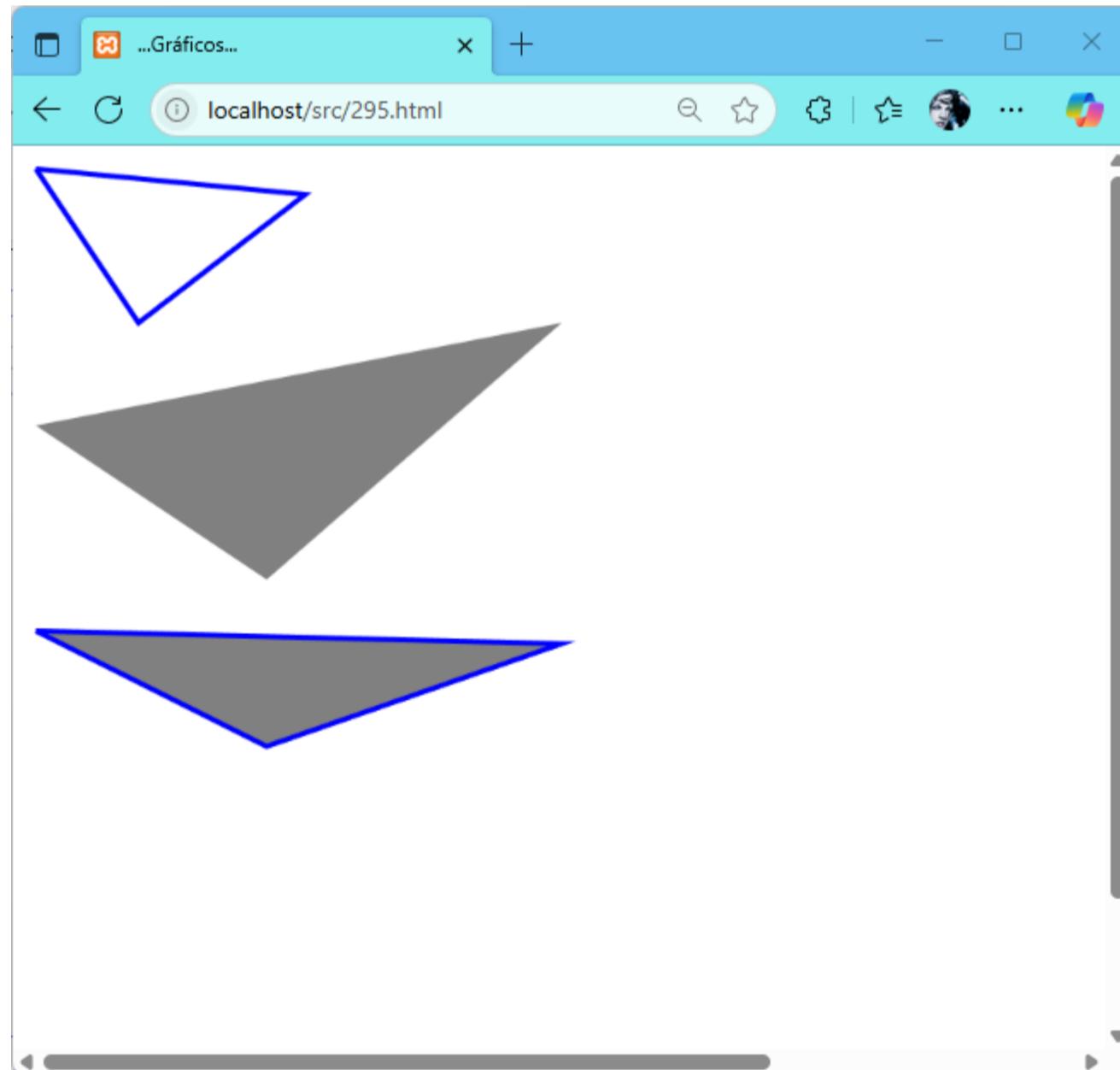


Ilustración 101: Uso de librería gráfica

```

<!DOCTYPE HTML><html lang="es">
<head><title>Gráficos</title></head>
<body>
<canvas id="milienzo" width="1200" height="800"></canvas>
<script src="LibreriaGrafica.js"></script>
<script>
    //Objeto para hacer los gráficos
    let Graficos = new Grafica(document.getElementById('milienzo'));

    //Lista de triángulos
    let Triangulos = [];

    //Genera triángulos al azar
    for (let cont = 1; cont <= 5; cont++) {
        let ColorRelleno = Graficos.ColorAzar(); //Algún color al azar
        let ColorBorde = Graficos.ColorAzar(); //Algún color al azar
        let GrosorBorde = Azar.EnteroEntre(1, 10); //Grueso entre 1 y 10

        //Coordenadas de las tres esquinas del triángulo
        let posX1 = Azar.EnteroEntre(1, 700); let posY1 = Azar.EnteroEntre(1, 700);
        let posX2 = Azar.EnteroEntre(1, 700); let posY2 = Azar.EnteroEntre(1, 700);
        let posX3 = Azar.EnteroEntre(1, 700); let posY3 = Azar.EnteroEntre(1, 700);

        //Tipo de triángulo: 1. Solo borde, 2. Sólo relleno, 3. Borde y relleno
        let Tipo = Azar.EnteroEntre(1, 3);

        Triangulos.push(new Triangulo(posX1, posY1, posX2, posY2, posX3, posY3, Tipo, GrosorBorde, ColorBorde,
        ColorRelleno));
    }

    Graficos.TrianguloLista(Triangulos);
</script>
</body>
</html>

```

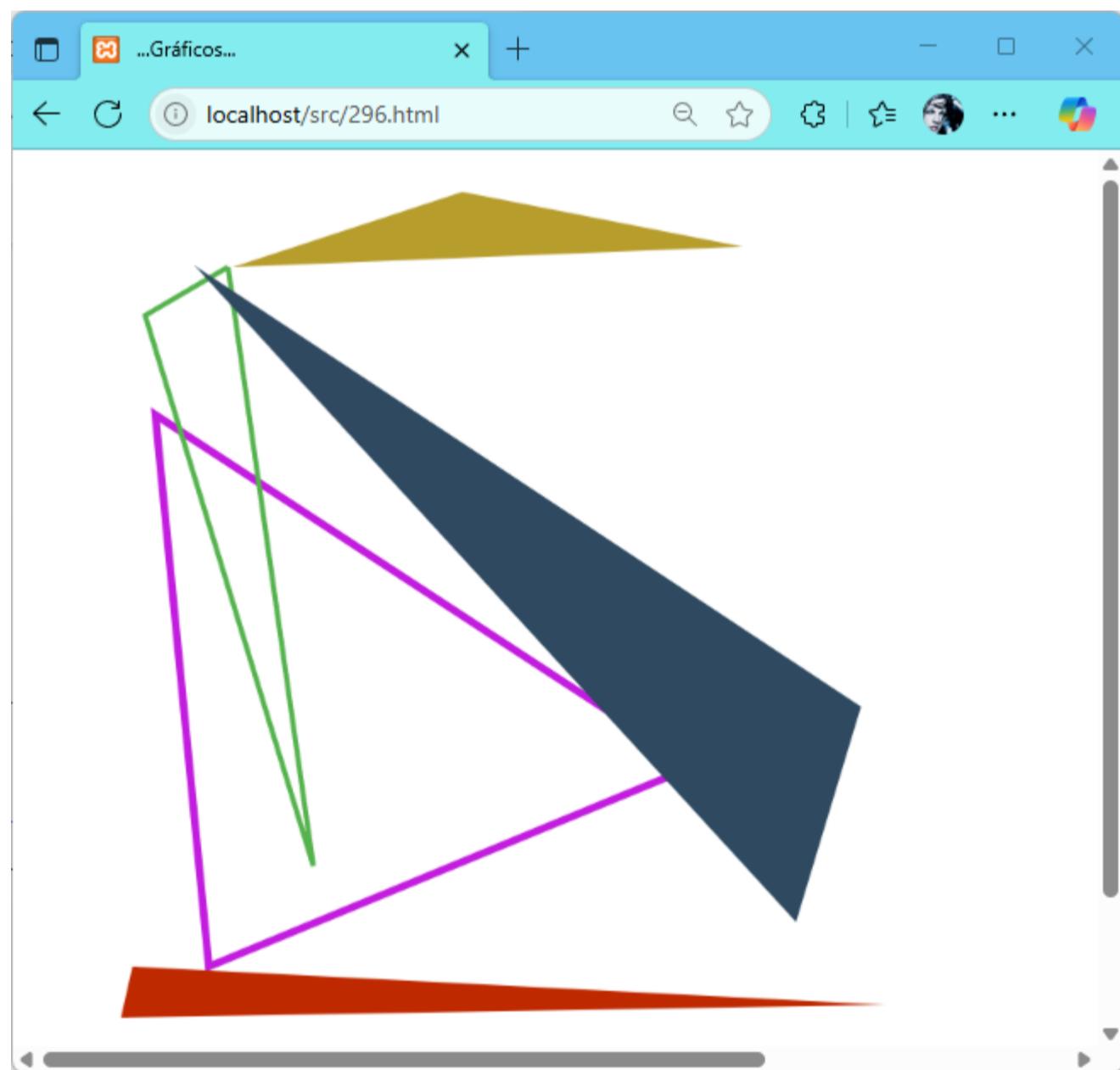


Ilustración 102: Uso de librería gráfica

```

<!DOCTYPE HTML><html lang="es">
<head><title>Gráficos</title></head>
<body>
<canvas id="milienzo" width="1200" height="800"></canvas>
<script src="LibreriaGrafica.js"></script>
<script>
    //Objeto para hacer los gráficos
    let Graficos = new Grafica(document.getElementById('milienzo'));

    //Dibuja un rectángulo: posX, posY, Ancho, Alto, Grosor, Color
    let ColorBorde = 'blue';
    let posX = 10, posY = 10;
    let Ancho = 90, Alto = 50;
    let Grosor = 4;
    Graficos.Rectangulo(posX, posY, Ancho, Alto, Grosor, ColorBorde);

    //Dibuja un rectángulo relleno: posX, posY, Ancho, Alto, ColorRelleno
    let ColorRelleno = 'gray';
    posX = 10; posY = 210;
    Graficos.RectanguloRelleno(posX, posY, Ancho, Alto, ColorRelleno);

    //Dibuja un rectángulo relleno con borde: posX, posY, Ancho, Alto, ColorRelleno, GrosorBorde, ColorBorde
    posX = 10; posY = 370;
    Graficos.RectanguloRellenoBorde(posX, posY, Ancho, Alto, ColorRelleno, Grosor, ColorBorde);

    //Un objeto de tipo rectángulo
    posX = 150;
    posY = 550;
    let Tipo = 3;
    ColorRelleno = Graficos.ColorAzar();
    ColorBorde = Graficos.ColorAzar();
    MiRectangulo = new Rectangulo(posX, posY, Ancho, Alto, Tipo, Grosor, ColorBorde, ColorRelleno);
    Graficos.RectanguloObj(MiRectangulo);
    MiRectangulo.Trasladar(400, 100);
    Graficos.RectanguloObj(MiRectangulo);
</script>
</body>
</html>

```

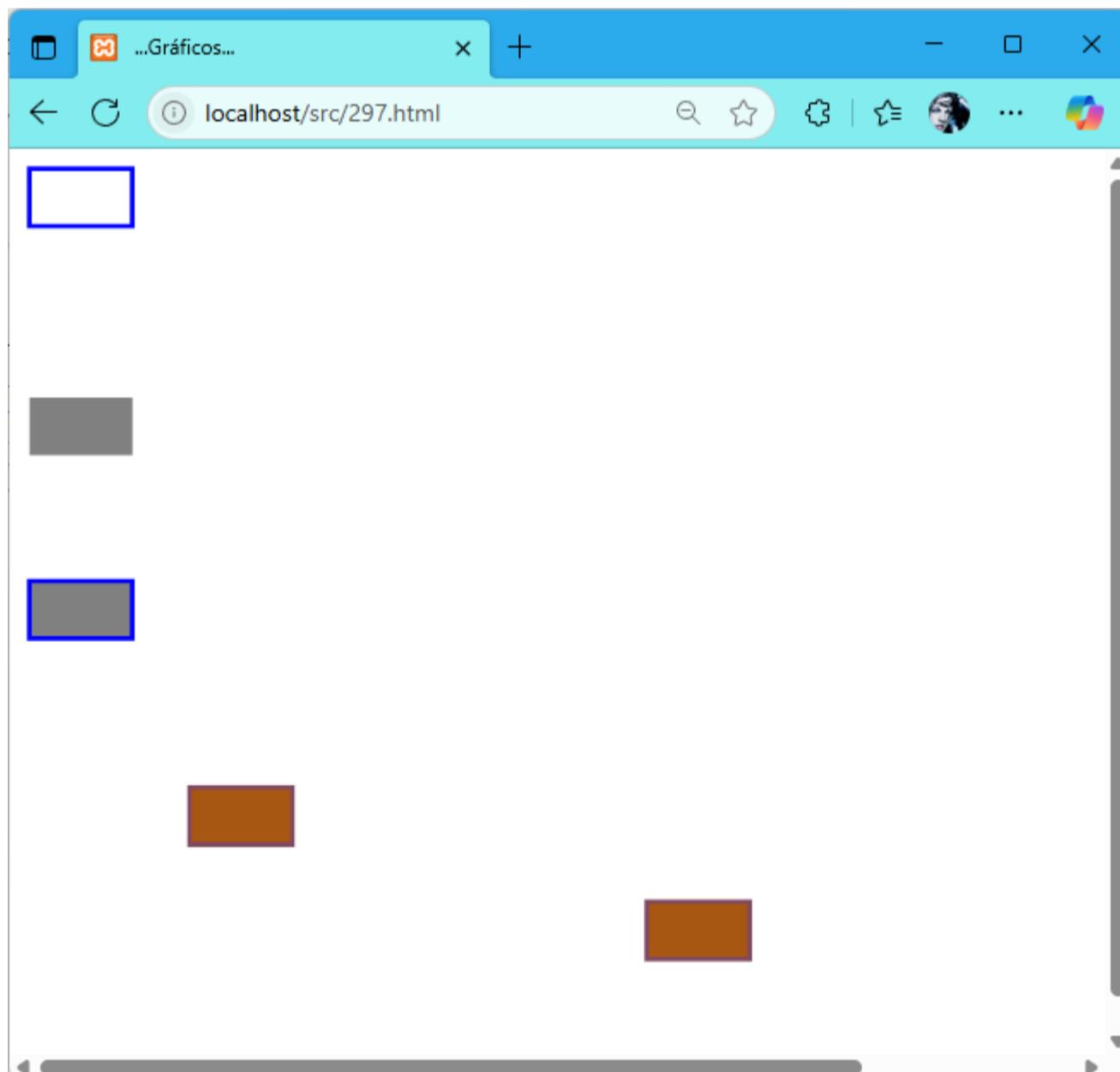


Ilustración 103: Uso de librería gráfica

De 298.html a 328.html hay diversos ejemplos de uso de la librería.

Referencias

- [1] W3Schools, «JavaScript Tutorial,» julio 2025. [En línea]. Available: <https://www.w3schools.com/js/>. [Último acceso: julio 2025].
- [2] Mozilla, «JavaScript,» julio 2025. [En línea]. Available: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. [Último acceso: julio 2025].
- [3] W3Techs, «Usage of client-side programming languages for websites,» julio 2025. [En línea]. Available: https://w3techs.com/technologies/overview/client_side_language/all. [Último acceso: julio 2025].
- [4] Adobe, «Flash,» diciembre 2020. [En línea]. Available: <https://www.adobe.com/co/products/flashplayer/end-of-life-alternative.html?msockid=0d2ac1506dfe6a15329cd7496c516b12>. [Último acceso: 07 julio 2025].
- [5] ECMA International, «ECMAScript® 2025 language specification,» junio 2025. [En línea]. Available: <https://ecma-international.org/publications-and-standards/standards/ecma-262/>. [Último acceso: 04 julio 2025].
- [6] W3Schools, «JavaScript versions,» 2025. [En línea]. Available: https://www.w3schools.com/js/js_2025.asp. [Último acceso: julio 2025].
- [7] Mozilla, «What is a web server?,» julio 2025. [En línea]. Available: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server. [Último acceso: julio 2025].
- [8] M. Rouse, «database management system (DBMS),» julio 2025. [En línea]. Available: <https://www.techtarget.com/searchdatamanagement/definition/database-management-system>. [Último acceso: julio 2025].
- [9] MariaDB, «MariaDB,» julio 2025. [En línea]. Available: <https://mariadb.org/>. [Último acceso: julio 2025].
- [10] MySQL, «MySQL,» julio 2025. [En línea]. Available: <https://www.mysql.com/>. [Último acceso: julio 2025].