

Autor: Rafael Alberto Moreno Parra

Capacitándose en JAVASCRIPT

2024

Contenido

| | |
|---|----|
| Tabla de Ilustraciones | 6 |
| Otros libros del autor | 11 |
| Página web del autor y canal en Youtube | 13 |
| Sitio en GitHub | 13 |
| Historial de actualizaciones | 13 |
| Licencia de este libro | 14 |
| Licencia del software | 14 |
| Marcas registradas..... | 14 |
| Introducción..... | 15 |
| Iniciando..... | 16 |
| Navegadores | 16 |
| Editores de texto..... | 16 |
| Código fuente..... | 16 |
| Comentarios en el código | 18 |
| El tradicional “Hola Mundo” | 19 |
| Uso de variables..... | 21 |
| Operaciones matemáticas | 24 |
| Asignación | 25 |
| Operaciones con cadenas | 26 |
| Impresión de caracteres propios del idioma español..... | 31 |
| Constantes matemáticas | 33 |
| Funciones trigonométricas | 34 |
| Funciones Matemáticas | 35 |
| Captura de datos tipo texto por pantalla | 36 |
| Captura de datos tipo numérico por pantalla | 37 |
| Ejemplos de cálculos | 38 |
| Corrigiendo errores en JavaScript..... | 39 |
| Herramienta de Google para programar en JavaScript..... | 41 |
| Validación en línea del código | 42 |
| Sí condicional | 45 |
| Operadores lógicos: Conjunción (Y, &&) y Disyunción (O,) | 47 |
| Uso del switch | 48 |
| Ciclos. Uso del “for” | 49 |
| Ciclo while | 54 |
| Ciclo do - while | 57 |
| Uso del break | 60 |
| Uso del continue | 61 |
| Ciclos anidados y salir de estos..... | 62 |
| Funciones o subrutinas | 64 |
| Ejemplos de uso de la librería | 70 |
| Método de Bisección | 80 |
| Ámbito (scope) de las variables y funciones..... | 81 |
| Funciones recursivas | 85 |
| Uso de eval como evaluador de expresiones algebraicas | 90 |
| Números aleatorios | 92 |
| Distribución Normal..... | 93 |
| Distribución Triangular..... | 94 |
| Distribución Uniforme | 95 |
| Manejo de errores con números | 96 |

| | |
|--|-----|
| isNaN | 96 |
| isFinite | 98 |
| Try...catch | 99 |
| Arreglos unidimensionales o vectores..... | 101 |
| Borrar elementos con splice() | 105 |
| Operaciones con arreglos | 106 |
| Ordenación de arreglo unidimensional | 109 |
| Funciones genéricas para arreglos | 112 |
| Retornar arreglos desde funciones..... | 114 |
| Implementación de algoritmos de ordenación | 115 |
| Algoritmo de burbuja..... | 115 |
| Algoritmo de ordenación por selección..... | 118 |
| Algoritmo de ordenación por inserción..... | 120 |
| Algoritmo de ordenación QuickSort | 122 |
| Algoritmo de ordenación MergeSort..... | 124 |
| Manejo de Fechas | 126 |
| Manejo de cadenas o strings | 129 |
| Ejemplos de programas usando cadenas | 135 |
| Invertir Cadena..... | 135 |
| Quitar los espacios de una cadena | 136 |
| Quitar las vocales de una cadena | 137 |
| Quitar caracteres no permitidos..... | 138 |
| Un sencillo cifrado / descifrado | 139 |
| Un sencillo cifrado / descifrado con clave de cifrado | 140 |
| Sumar largos números enteros almacenados en cadenas | 142 |
| Arreglos bidimensionales..... | 143 |
| Ejemplos de programas usando arreglos bidimensionales | 147 |
| Poner una torre al azar en un tablero de ajedrez y mostrar su ataque | 147 |
| Poner un Rey al azar en un tablero de ajedrez y mostrar su ataque | 148 |
| Poner un alfil al azar en un tablero de ajedrez y mostrar su ataque..... | 149 |
| Poner una Reina al azar en un tablero de ajedrez y mostrar su ataque | 151 |
| Ruta del menor costo..... | 153 |
| Resolver Sudokus..... | 155 |
| Poner los barcos en el juego batalla naval | 157 |
| Funciones que reciben distinto número de parámetros | 159 |
| Funciones que reciben a su vez funciones por parámetros | 161 |
| Operaciones de bit..... | 163 |
| Convertir un entero a su representación binaria | 163 |
| Operaciones OR, AND y XOR directas..... | 164 |
| Multiplicar usando operaciones de bit | 167 |
| Dividir usando operaciones de bit | 168 |
| Intercambiar valores usando operaciones de bit | 169 |
| Programación orientada a objetos | 170 |
| Definiendo clases, constructores y atributos | 170 |
| Definiendo métodos | 171 |
| Métodos y atributos | 172 |
| Atributos y métodos privados | 173 |
| Getters y Setters | 174 |
| Herencia | 176 |
| Herencia y constructor..... | 177 |

| | |
|--|-----|
| Sobrecarga de métodos | 179 |
| Uso de typeof para detectar el tipo de dato de una variable | 180 |
| Árbol binario | 182 |
| Recorrido en pre-orden | 183 |
| Recorrido en in-orden | 184 |
| Recorrido en post-orden | 185 |
| Ordenación usando un árbol binario | 186 |
| Lista de objetos. Pilas | 187 |
| Lista de objetos | 189 |
| Instrucciones que ejecutan controlando el tiempo | 191 |
| Uso de JSON | 193 |
| Control sobre la página HTML | 201 |
| Captura el evento de dar clic en un <div> | 203 |
| Captura el evento de dar doble clic en un <div> | 205 |
| Captura el evento de pasar el cursor del ratón por encima de un <div> | 206 |
| Captura el evento de mover el cursor del ratón fuera del <div> | 207 |
| Captura el evento apenas cargue la página | 208 |
| Captura el evento cuando de clic al interior del <div> | 209 |
| Captura el evento al soltar el botón del ratón dentro del <div> | 210 |
| Captura el evento de mover el cursor del ratón | 211 |
| Captura el evento cuando se cambia el tamaño de la ventana | 212 |
| Captura el evento de dar “submit” en un formulario | 213 |
| Captura el evento de entrar a la caja de texto | 214 |
| Captura el evento al salir de una caja de texto | 215 |
| Captura el evento al presionar una tecla dentro de una caja de texto | 216 |
| Captura el evento cuando suelta la tecla | 218 |
| Captura el evento de cambio del texto en una caja de texto | 219 |
| Mostrar el código de la tecla presionada | 220 |
| Preguntar antes de ir a la página enlazada | 221 |
| DOM (Document Object Model) | 222 |
| Cambiar atributos de un objeto | 222 |
| Cambiar el color de fondo de un <div> | 223 |
| Hacer visible o no un <div> | 224 |
| Poner o quitar texto de un <div> | 225 |
| Cambiar el aspecto del objeto cuando está seleccionado | 226 |
| Validar valores al saltar a otro objeto | 228 |
| Contar objetos | 229 |
| Obtener lista de objetos | 230 |
| Obtener determinado objeto de una lista | 231 |
| Obtener lista de determinado tipo de objetos y ver el valor de sus atributos | 232 |
| Obtener la lista de determinado tipo de objetos y cambiar sus atributos | 233 |
| Mostrar la ubicación del documento | 234 |
| Cambiar el estilo CSS de un determinado objeto | 235 |
| Mostrar la codificación del documento | 236 |
| Mostrar el título del documento | 237 |
| Saltos de línea sin usar | 238 |
| Mostrar la fecha de modificación del documento | 239 |
| Mostrar el tipo de documento | 240 |
| Mostrar el dominio del documento | 241 |
| Mostrar el estado del documento | 242 |

| | |
|---|-----|
| Mostrar las dimensiones del documento | 243 |
| Mostrar el número de enlaces..... | 244 |
| Mostrar hacia donde apuntan los enlaces..... | 245 |
| Crear botones en tiempo de ejecución | 247 |
| Crear <p> en tiempo de ejecución..... | 248 |
| Crear nuevos <p> y contarlos en tiempo de ejecución | 249 |
| Contar el número de objetos en el documento | 250 |
| Contar el número de formularios en el documento..... | 251 |
| Contar el número de imágenes del documento | 252 |
| Describir los objetos de la página | 253 |
| Abrir una ventana | 255 |
| Abrir un conjunto de ventanas | 256 |
| Un formulario que ejecuta un algoritmo local al enviar la información | 257 |
| Valida un correo con una expresión regular..... | 259 |
| Valida una URL con expresiones regulares | 260 |
| Gráficos | 261 |
| Dibujar líneas | 263 |
| Cambiar el grueso de una línea | 264 |
| Dar un color determinado a una línea..... | 265 |
| Color RGB | 266 |
| Estilos para finalizar las líneas..... | 267 |
| Dibujar varias líneas..... | 268 |
| Juntar líneas | 269 |
| Probando los otros estilos de juntar líneas | 270 |
| Varias líneas | 271 |
| Genera una figura cerrada y la rellena | 272 |
| Dibujar varias líneas horizontales con un ciclo | 273 |
| Dibujar varias líneas verticales con un ciclo | 274 |
| Ilusión de curva con varias líneas | 275 |
| Dibujar círculos | 277 |
| Dibujar curvas | 279 |
| Curva de Bézier | 280 |
| Combinando curvas | 281 |
| Rectángulo y gradiente de color | 282 |
| Variando en forma oblicua con varios colores | 283 |
| Variando en forma vertical | 284 |
| Variando en forma horizontal..... | 285 |
| Variando en círculos..... | 286 |
| Trabajando con imágenes..... | 287 |
| Variar posición y tamaño de la imagen | 288 |
| Convertir imagen a escala de grises | 289 |
| Dibujar letras..... | 291 |
| Sombras | 296 |
| Semitransparente | 297 |
| Referencias..... | 298 |

Tabla de Ilustraciones

| | |
|--|----|
| Ilustración 1: Lenguajes de programación para aplicaciones Web en el lado del cliente..... | 15 |
| Ilustración 2: Código fuente en GitHub | 16 |
| Ilustración 3: Notepad++ en español..... | 16 |
| Ilustración 4: Configuramos Notepad++ para editar archivos HTML y JavaScript | 17 |
| Ilustración 7: Botón de guardar | 19 |
| Ilustración 8: Guardar la página HTML en el PC | 19 |
| Ilustración 9: Debemos fijarnos que el archivo tenga extensión .html | 19 |
| Ilustración 12: Mensaje en negrillas | 20 |
| Ilustración 13: Impresión del valor de cada variable | 21 |
| Ilustración 14: Impresión de valor de cada variable..... | 22 |
| Ilustración 15: Impresión de valor de cada variable..... | 23 |
| Ilustración 16: Resultado de la ejecución de las operaciones matemáticas | 24 |
| Ilustración 17: Combinando texto con valores..... | 25 |
| Ilustración 18: Resultado de la concatenación | 26 |
| Ilustración 19: Se concatena texto con números | 27 |
| Ilustración 20: Se concatena números y texto | 28 |
| Ilustración 21: Primero se hace la operación matemática y luego la concatenación | 29 |
| Ilustración 22: Se hace la operación matemática y el resto es concatenación | 30 |
| Ilustración 23: Imprimir caracteres propios del idioma español..... | 31 |
| Ilustración 24: En Chrome puede pasar que no se impriman | 31 |
| Ilustración 25: Funciona en Mozilla Firefox | 31 |
| Ilustración 26: Ante un fallo así, verificamos la codificación. Usualmente está en "Codificar en ANSI" | 31 |
| Ilustración 27: Seleccionamos la opción de "Convertir a UTF-8" | 32 |
| Ilustración 28: Después de guardar, chequeamos que esté en la opción "Codificar en UTF-8" | 32 |
| Ilustración 29: Constantes matemáticas | 33 |
| Ilustración 30: Funciones trigonométricas | 34 |
| Ilustración 31: Funciones matemáticas | 35 |
| Ilustración 32: Cuadro de diálogo para capturar un valor texto | 36 |
| Ilustración 33: Muestra el valor capturado | 36 |
| Ilustración 34: Cuadro de diálogo que capture un valor texto | 37 |
| Ilustración 35: Capturando un valor texto que después se convertirá en real | 37 |
| Ilustración 36: Resultado de la operación matemática | 37 |
| Ilustración 37: Código en JavaScript con varios errores | 39 |
| Ilustración 38: Ante errores en JavaScript, el navegador queda en blanco | 39 |
| Ilustración 39: Presionando F12 en el navegador Edge y dando clic en Consola se muestra el error | 40 |
| Ilustración 40: Closure Compiler para optimizar código en JavaScript | 41 |
| Ilustración 41: JavaScript Validator para chequear sintaxis | 42 |
| Ilustración 42: Diagnóstico de JavaScript Validator de un script de ejemplo | 43 |
| Ilustración 43:JavaScript Tester para validar sintaxis | 43 |
| Ilustración 44: Jshint para validar sintaxis | 44 |
| Ilustración 45: Ejecución del ciclo ascendente | 49 |
| Ilustración 46: Ejecución del ciclo descendente | 49 |
| Ilustración 47: Ciclo ascendente de 3 en 3 | 50 |
| Ilustración 48: Ciclo ascendente de 10 en 10 | 50 |
| Ilustración 49: Ciclo ascendente doblando el anterior | 51 |
| Ilustración 50: Ciclo descendente dividiendo entre 2 el anterior | 51 |
| Ilustración 51: Ciclo que muestra el progreso de dos variables | 52 |
| Ilustración 52: Ciclo que muestra el progreso de dos variables: | 53 |
| Ilustración 53: Ciclos anidados y salir de estos..... | 62 |
| Ilustración 54: Ciclos anidados y salir de estos..... | 63 |
| Ilustración 55: Funciones, subrutinas o procedimientos..... | 64 |
| Ilustración 56: Entre 1000 y 9999. Cantidad de números que la suma de las tres últimas cifras es par: 4500 | 70 |
| Ilustración 57: Entre 1000 y 9999. Cantidad de números que la multiplicación de las tres últimas cifras es igual a la suma de esas tres últimas cifras: 63 | 71 |
| Ilustración 58: Entre 8000 y 9000. Cantidad de números que cada una de sus tres últimas cifras es menor de 5: 126 | 72 |
| Ilustración 59: Entre 8500 y 8700. Cantidad de números que al juntar la antepenúltima cifra y la última cifra se obtenga un primo: 40 | 73 |
| Ilustración 60: Entre 4000 y 7000. Cantidad de números primos que no tengan el número 3 en sus cifras: 215 | 74 |
| Ilustración 61: Entre 1000 y 9999. Cantidad de números palíndromos que tengan mínimo dos veces el 7 en sus cifras: 18 | 75 |
| Ilustración 62: Entre 5000 y 9000. Cantidad de números que tengan solo cifras pares y al menos un número 4: 122 | 76 |
| Ilustración 63: Entre 7700 y 8000. Cantidad de números que la suma de sus cifras es impar y al menos tenga una vez el número 7: 150 | 77 |
| Ilustración 64: Entre 1000 y 9999. Cantidad de números primos que sus cifras están entre 3 y 7: 72 | 78 |
| Ilustración 65: Entre 10 y 999999. Cantidad de números primos y a su vez palíndromos: 109 | 79 |
| Ilustración 66: Método de bisección | 80 |
| Ilustración 67: Ámbito (scope) de una variable | 81 |

| | |
|---|-----|
| Ilustración 68: Ámbito (scope) de una variable..... | 82 |
| Ilustración 69: Ámbito (scope) de una variable..... | 83 |
| Ilustración 70: Ámbito (scope) de una variable..... | 84 |
| Ilustración 71: Cálculo de la factorial con función recursiva | 85 |
| Ilustración 72: Cálculo del máximo común divisor con función recursiva | 86 |
| Ilustración 73: Suma de cifras con función recursiva | 87 |
| Ilustración 74: Conversión a número binario con función recursiva..... | 88 |
| Ilustración 75: Cálculo de potencia con función recursiva | 89 |
| Ilustración 76: Eval como evaluador de expresiones | 90 |
| Ilustración 77: Eval como evaluador de expresiones | 90 |
| Ilustración 78: Eval como evaluador de expresiones | 91 |
| Ilustración 79: Número aleatorio..... | 92 |
| Ilustración 80: Listado de números aleatorios | 92 |
| Ilustración 81: Distribución normal | 93 |
| Ilustración 82: Distribución triangular | 94 |
| Ilustración 83: Distribución uniforme | 95 |
| Ilustración 84: Uso de isNaN..... | 97 |
| Ilustración 85: Ingresa dividendo..... | 98 |
| Ilustración 86: Ingresa divisor como cero para generar un error..... | 98 |
| Ilustración 87: La división entre cero da "Infinity" | 98 |
| Ilustración 88: Ingresa expresión válida | 99 |
| Ilustración 89: Hace el cálculo con expresión válida | 99 |
| Ilustración 90: Ingresa expresión con sintaxis errónea | 99 |
| Ilustración 91: Mensaje de error por la sintaxis errónea | 100 |
| Ilustración 92: Arreglo unidimensional impreso | 101 |
| Ilustración 93: Muestra ítem a ítem el arreglo unidimensional | 102 |
| Ilustración 94: Otra forma de declarar arreglo unidimensional..... | 103 |
| Ilustración 95: Otra forma de declarar arreglo unidimensional..... | 104 |
| Ilustración 96: Uso de splice() para borrar ítems en un arreglo | 105 |
| Ilustración 97: Operación matemática con elementos del arreglo unidimensional | 106 |
| Ilustración 98: Operación matemática con elementos del arreglo unidimensional | 107 |
| Ilustración 99: Operación matemática con elementos del arreglo unidimensional | 108 |
| Ilustración 100: Ordenación de arreglo unidimensional | 109 |
| Ilustración 101: Ordenación de arreglo unidimensional | 110 |
| Ilustración 102: Ordenación de arreglo de menor a mayor, y de mayor a menor..... | 111 |
| Ilustración 103: Dado un arreglo extraer los valores únicos | 112 |
| Ilustración 104: ¿Función genérica para hallar menor y mayor valor? | 113 |
| Ilustración 105: Separe los elementos pares e impares de una lista, y retorne ambas listas..... | 114 |
| Ilustración 106: Ordenación por el método de burbuja..... | 115 |
| Ilustración 107: Ordenación por burbuja | 116 |
| Ilustración 108: Ordenación por método de burbuja mejorado | 117 |
| Ilustración 109: Fecha actual | 126 |
| Ilustración 110: Fecha actual | 127 |
| Ilustración 111: Fecha actual | 128 |
| Ilustración 112: Concatenación de cadenas | 129 |
| Ilustración 113: Concatenación de números y cadenas | 130 |
| Ilustración 114: Primera letra de una cadena..... | 131 |
| Ilustración 115: Longitud de una cadena..... | 132 |
| Ilustración 116: Convertir a mayúsculas y minúsculas | 133 |
| Ilustración 117: Comparación de cadenas..... | 134 |
| Ilustración 118: Invierte la cadena..... | 135 |
| Ilustración 119: Quita los espacios de una cadena..... | 136 |
| Ilustración 120: Quitar vocales (minúsculas, mayúsculas, tildadas) | 137 |
| Ilustración 121: Quitar vocales (minúsculas, mayúsculas, tildadas) | 137 |
| Ilustración 122: Retirar caracteres no permitidos de una cadena | 138 |
| Ilustración 123: Cifrado sencillo..... | 139 |
| Ilustración 124: Cifrado sencillo..... | 139 |
| Ilustración 125: Cifrado sencillo con clave de cifrado | 140 |
| Ilustración 126: Cifrado sencillo con clave de cifrado | 141 |
| Ilustración 127: Cifrado sencillo con clave de cifrado | 141 |
| Ilustración 128: Suma dos números enteros positivos almacenados en cadenas | 142 |
| Ilustración 129: Crea el arreglo bidimensional | 143 |
| Ilustración 130: Pone una X en una posición al azar | 144 |
| Ilustración 131: Función genérica para crear arreglo bidimensional | 145 |

| | |
|--|-----|
| Ilustración 132: Función genérica para crear arreglo bidimensional | 146 |
| Ilustración 133: Poner en una posición al azar la torre y mostrar su ataque..... | 147 |
| Ilustración 134: Poner un Rey al azar en un tablero de ajedrez y mostrar su ataque | 148 |
| Ilustración 135: Poner un alfil al azar en un tablero de ajedrez y mostrar su ataque | 150 |
| Ilustración 136: Poner un alfil al azar en un tablero de ajedrez y mostrar su ataque | 150 |
| Ilustración 137: Poner una Reina al azar en un tablero de ajedrez y mostrar su ataque | 152 |
| Ilustración 138: Poner una Reina al azar en un tablero de ajedrez y mostrar su ataque | 152 |
| Ilustración 139: Ruta del menor costo. Genera nuevas rutas más económicas | 154 |
| Ilustración 140: Sudoku resuelto | 156 |
| Ilustración 141: Barcos ubicados en el juego de mesa Batalla Naval..... | 158 |
| Ilustración 142: Número de parámetros variable en funciones..... | 159 |
| Ilustración 143: Muestra el contenido de cada parámetro de la función | 160 |
| Ilustración 144: Envía una función como parámetro a otra función..... | 161 |
| Ilustración 145: Envía una función como parámetro a otra función..... | 162 |
| Ilustración 146: Convertir un entero a su representación binaria | 163 |
| Ilustración 147: Operación OR..... | 164 |
| Ilustración 148: Operación AND | 165 |
| Ilustración 149: Operación XOR..... | 166 |
| Ilustración 150: Multiplicación usando operaciones de bit..... | 167 |
| Ilustración 151: División usando operaciones de bit | 168 |
| Ilustración 152: Intercambiar valores usando operaciones de bit | 169 |
| Ilustración 153: Valores de atributos..... | 170 |
| Ilustración 154: Métodos | 171 |
| Ilustración 155: Métodos y atributos | 172 |
| Ilustración 156: Métodos privados | 173 |
| Ilustración 157: Getters y Setters | 174 |
| Ilustración 158: Getters y Setters | 175 |
| Ilustración 159: Herencia | 176 |
| Ilustración 160: Herencia y constructor..... | 177 |
| Ilustración 161: Herencia y constructor..... | 178 |
| Ilustración 162: No hay sobrecarga de métodos en JavaScript..... | 179 |
| Ilustración 163: typeof | 180 |
| Ilustración 164: typeof de objetos..... | 181 |
| Ilustración 165: Recorrido de árbol binario..... | 182 |
| Ilustración 166: Árbol binario | 183 |
| Ilustración 167: Recorrido en pre-orden | 183 |
| Ilustración 168: Recorrido en in-orden..... | 184 |
| Ilustración 169: Recorrido en post-orden..... | 185 |
| Ilustración 170: Ordenación usando un árbol binario | 186 |
| Ilustración 171: Lista de objetos. Pilas..... | 187 |
| Ilustración 172: Lista de objetos. Pilas..... | 188 |
| Ilustración 173: Primer ejemplo de uso de JSON | 193 |
| Ilustración 174: Segundo ejemplo de uso de JSON | 194 |
| Ilustración 175: Tercer ejemplo de uso de JSON | 195 |
| Ilustración 176: Cuarto ejemplo de uso de JSON | 196 |
| Ilustración 177: Quinto ejemplo de uso de JSON | 197 |
| Ilustración 178: Sexto ejemplo de uso de JSON | 198 |
| Ilustración 179: Séptimo ejemplo de uso de JSON | 199 |
| Ilustración 180: Octavo ejemplo de uso de JSON | 200 |
| Ilustración 181: Botón..... | 201 |
| Ilustración 182: Ventana emergente generada por JavaScript | 201 |
| Ilustración 183: Ventana emergente al dar clic en un hipervínculo..... | 202 |
| Ilustración 184: Ventana emergente al dar clic en un <div> | 203 |
| Ilustración 185: Ventana emergente al dar clic en un <div> | 204 |
| Ilustración 186: Ventana emergente al dar doble clic en un <div> | 205 |
| Ilustración 187: Ventana emergente al mover el cursor por encima de un <div> | 206 |
| Ilustración 188: Ventana emergente al mover el cursor fuera de un <div>..... | 207 |
| Ilustración 189: Ventana emergente al entrar a una página..... | 208 |
| Ilustración 190: Dar clic al interior de un <div> | 209 |
| Ilustración 191: Soltar la tecla izquierda del ratón en el interior de un <div> | 210 |
| Ilustración 192: Cambiar aleatoriamente el color de fondo al mover el cursor del ratón al interior de un <div> | 211 |
| Ilustración 193: Mostrar un aviso cuando se cambia el tamaño de la ventana del navegador | 212 |
| Ilustración 194: Mensaje cuando se presiona botón de envío de formulario | 213 |
| Ilustración 195: Mensaje cuando accede a una caja de texto..... | 214 |

| | |
|---|-----|
| Ilustración 196: Mensaje cuando se sale de una caja de texto | 215 |
| Ilustración 197: Mensaje cuando se presiona cualquier tecla dentro de la caja de texto | 216 |
| Ilustración 198: Mensaje cuando deja de presionar una tecla en una caja de texto..... | 218 |
| Ilustración 199: Mensaje cuando se ha hecho un cambio dentro de una caja de texto..... | 219 |
| Ilustración 200: Mostrar el código ASCII de la tecla presionada | 220 |
| Ilustración 201: Intercepta el poder navegar a otra página | 221 |
| Ilustración 202: Una etiqueta común | 222 |
| Ilustración 203: La etiqueta común se convierte en hipervínculo | 222 |
| Ilustración 204: Un botón | 223 |
| Ilustración 205: Se cambia el color de fondo de un <div> al presionar el botón | 223 |
| Ilustración 206: Un <div> y dos botones..... | 224 |
| Ilustración 207: Un botón vuelve invisible el <div> | 224 |
| Ilustración 208: Un <div> con texto y dos botones | 225 |
| Ilustración 209: Al presionar el botón de "Borrar" se borra el texto del <div>..... | 225 |
| Ilustración 210: Cuando la caja de texto está seleccionada tiene un estilo | 226 |
| Ilustración 211: Cuando la caja de texto pierde el foco, tendrá otro estilo | 226 |
| Ilustración 212: Muestra con determinado color si el dato ingresado no es un número válido | 228 |
| Ilustración 213: Si es un número válido, entonces deja el color por defecto | 228 |
| Ilustración 214: Cuenta los objetos con la etiqueta <p> | 229 |
| Ilustración 215: Captura un determinado objeto con <p> y lo examina | 230 |
| Ilustración 216: Obtiene la lista de objetos tipo entrada, trae el primero y muestra su valor | 231 |
| Ilustración 217: Muestra el valor que tiene cada objeto <p>..... | 232 |
| Ilustración 218:Párrafos <p> con estilo por defecto | 233 |
| Ilustración 219: Se cambia el estilo de cada objeto <p> | 233 |
| Ilustración 220: Muestra la URL donde se encuentra la página | 234 |
| Ilustración 221: Dos objetos <p> | 235 |
| Ilustración 222: Se pueden cambiar los atributos de un objeto <p> en particular | 235 |
| Ilustración 223: Muestra la codificación que tiene la página | 236 |
| Ilustración 224: Muestra el título que tiene la página..... | 237 |
| Ilustración 225: Saltos de línea sin usar | 238 |
| Ilustración 226: Muestra las fechas de modificación de la página | 239 |
| Ilustración 227: Muestra el tipo de documento | 240 |
| Ilustración 228: Mostrar el dominio donde está alojada la página | 241 |
| Ilustración 229: Muestra el estado del documento. Si ha cargado completamente es "complete" | 242 |
| Ilustración 230: Alto y ancho de la página | 243 |
| Ilustración 231: Alto y ancho de la página adicionando más <p> | 243 |
| Ilustración 232: Una página con dos enlaces..... | 244 |
| Ilustración 233: Cuenta el número de enlaces de la página..... | 244 |
| Ilustración 234: Una página con tres enlaces | 245 |
| Ilustración 235: Muestra el enlace de cada página | 245 |
| Ilustración 236: Muestra el enlace de cada página | 245 |
| Ilustración 237: Muestra el enlace de cada página | 246 |
| Ilustración 238: Un documento con un botón..... | 247 |
| Ilustración 239: Al presionar el botón se genera un nuevo botón | 247 |
| Ilustración 240: Y se pueden generar más botones en tiempo de ejecución | 247 |
| Ilustración 241: Un documento con un botón..... | 248 |
| Ilustración 242: Al presionar el botón se genera un nuevo <p>..... | 248 |
| Ilustración 243: Se generan nuevos <p> en tiempo de ejecución | 248 |
| Ilustración 244: Un documento con dos botones | 249 |
| Ilustración 245: Se generan dos <p> en tiempo de ejecución | 249 |
| Ilustración 246: Se cuenta el número de <p> en el documento..... | 249 |
| Ilustración 247: Varios tipos de objeto en la página | 250 |
| Ilustración 248: Cuenta el número de objetos en el documento..... | 250 |
| Ilustración 249: Cuenta el número de formularios en el documento | 251 |
| Ilustración 250: Un documento con una imagen | 252 |
| Ilustración 251: Cuenta el número de imágenes en el documento | 252 |
| Ilustración 252: Cuenta el número de atributos que tenga un objeto | 253 |
| Ilustración 253: Trae la identificación de un objeto | 253 |
| Ilustración 254: Trae el valor de la identificación de ese atributo | 254 |
| Ilustración 255: Trae otro atributo | 254 |
| Ilustración 256: Muestra el valor de ese segundo atributo..... | 254 |
| Ilustración 257: Abre una nueva ventana y apunta a una determinada dirección | 255 |
| Ilustración 258: Abre varias ventanas, cada una hacia una dirección distinta..... | 256 |
| Ilustración 259: Un formulario que ejecuta un algoritmo local al enviar la información | 257 |

| | |
|---|-----|
| Ilustración 260: Valida el correo electrónico | 259 |
| Ilustración 261: Valida el correo electrónico | 259 |
| Ilustración 262: Valida la URL | 260 |
| Ilustración 263: Valida la URL | 260 |
| Ilustración 264:Lienzo ("canvas") visible usando un borde | 262 |
| Ilustración 265: Línea dibujada | 263 |
| Ilustración 266: Línea gruesa | 264 |
| Ilustración 267: Color a la línea | 265 |
| Ilustración 268: Línea con color modificando los parámetros RGB | 266 |
| Ilustración 269: Las líneas finalizan redondeadas en ambos extremos usando "round" | 267 |
| Ilustración 270:Tres líneas con la misma longitud, pero diferente cierre de extremo | 268 |
| Ilustración 271: Dos líneas juntas con un empalme redondeado | 269 |
| Ilustración 272: Diferentes formas de juntar dos líneas: "round", "miter", "bevel" | 270 |
| Ilustración 273: Una figura cerrada | 271 |
| Ilustración 274: Figura cerrada y rellena de un color | 272 |
| Ilustración 275: Líneas generadas por un ciclo | 273 |
| Ilustración 276: Líneas verticales dibujadas con un ciclo | 274 |
| Ilustración 277: Ilusión de curva usando líneas rectas | 275 |
| Ilustración 278: Ilusión de dos curvas usando líneas rectas | 276 |
| Ilustración 279: La manera en que se dibujará el círculo | 277 |
| Ilustración 280: Círculo | 278 |
| Ilustración 281: Forma en que se dibujarán las curvas | 279 |
| Ilustración 282: Curva | 279 |
| Ilustración 283: Curva Bézier | 280 |
| Ilustración 284: Combinando curvas | 281 |
| Ilustración 285: Variación del color | 282 |
| Ilustración 286: Variación de varios colores en forma oblicua | 283 |
| Ilustración 287: Variación de color en forma vertical | 284 |
| Ilustración 288: Variación de color horizontal | 285 |
| Ilustración 289: Variación en círculos | 286 |
| Ilustración 290: La imagen rellena el rectángulo repitiéndose | 287 |
| Ilustración 291: Imagen original | 289 |
| Ilustración 292: Imagen en escala de grises | 290 |
| Ilustración 293: Dibuja un texto | 291 |
| Ilustración 294: Dibuja un texto con relleno de color | 292 |
| Ilustración 295: Texto dibujado, se muestra el borde de cada letra | 293 |
| Ilustración 296: Dibuja un texto alineado al final | 294 |
| Ilustración 297: Muestra el ancho del texto dibujado | 295 |
| Ilustración 298: Sombra | 296 |
| Ilustración 299: Rectángulo semitransparente | 297 |

Otros libros del autor

Libro 20: "Gráficos en C#". En Colombia 2022. Págs. 213. Libro y código fuente descargable en:
<https://github.com/ramsoftware/C-Sharp-Graficos>

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>

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

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>

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

Libro 14: "Iniciando en C#". En Colombia 2020. Págs. 72. Libro y código fuente descargable en:
<https://github.com/ramsoftware/C-Sharp-Iniciando>

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

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

Libro 11: "Capacitándose en JavaScript". En Colombia 2020. Págs. 317. **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. Estado: Obsoleto (No hay enlace).

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

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

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

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

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

Página web del autor y canal en Youtube

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

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

Sitio en GitHub

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

Historial de actualizaciones

| Fecha | Cambios |
|---------------------|-----------------------------------|
| 06 de enero de 2024 | Inicio del libro con 298 páginas. |

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 enero de 2024 de cómo está repartido los lenguajes de programación en el cliente [3]:

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

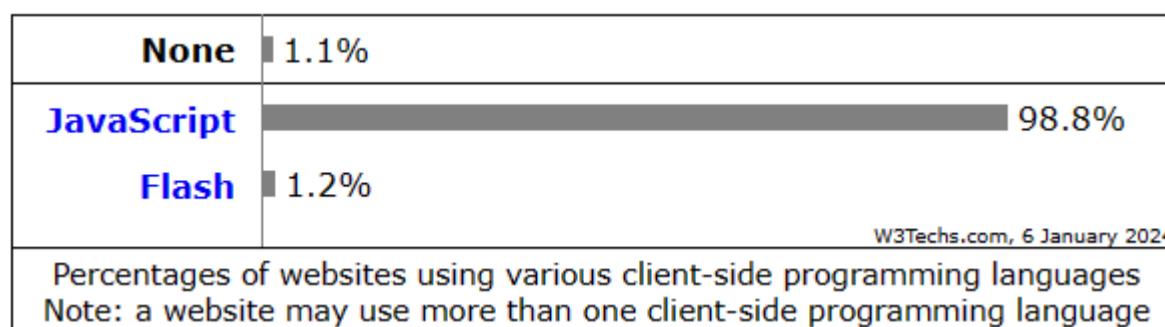


Ilustración 1: Lenguajes de programación para aplicaciones Web en el lado del cliente

JavaScript es el lenguaje de facto, además, le sumamos 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 2023 [5]

Iniciando

Empezar a programar en JavaScript es fácil y gratuito. Sólo requerimos tener un navegador moderno instalado en el PC (Microsoft Edge, Mozilla Firefox, Opera, Google Chrome, 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_2023.asp [6]

Editores de texto

Enlace para Notepad++: <https://notepad-plus-plus.org/downloads/v8.6/>

Código fuente

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

Ilustración 2: Código fuente en GitHub

Donde se muestre código fuente en JavaScript, se mostrará la palabra “Archivo N.html”. Significa que el código fuente de JavaScript está en el interior de un archivo HTML (en algunos casos scripts el archivo fuente está en un archivo aparte .JS).

Para escribir código fuente abrimos Notepad++, obtenemos un documento en blanco:

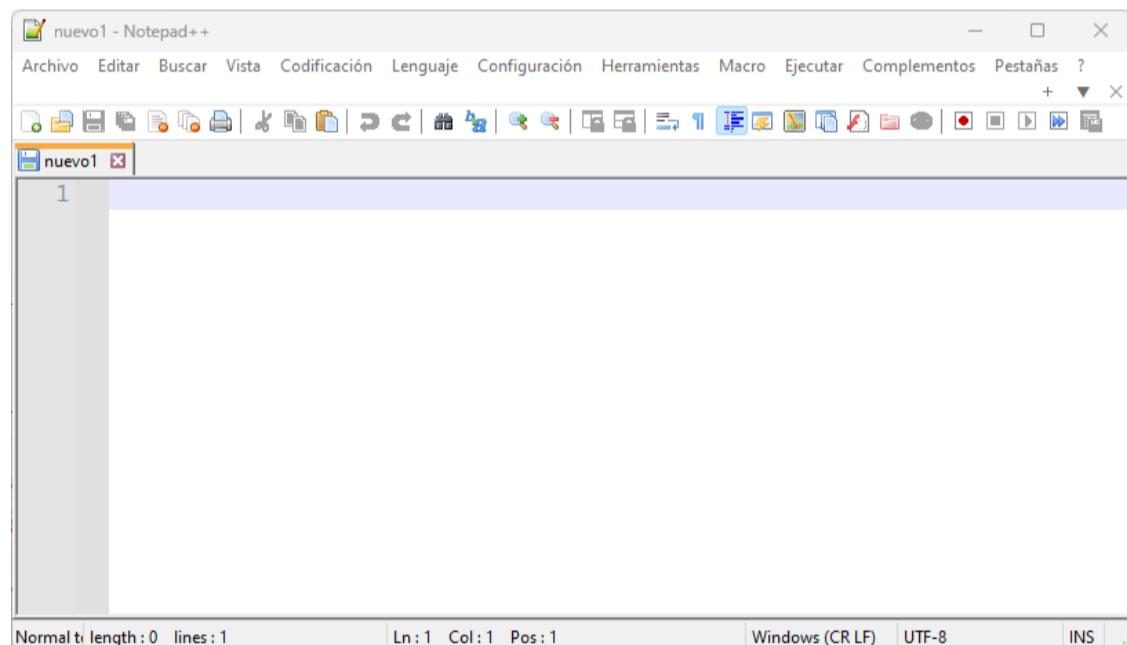


Ilustración 3: Notepad++ en español

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

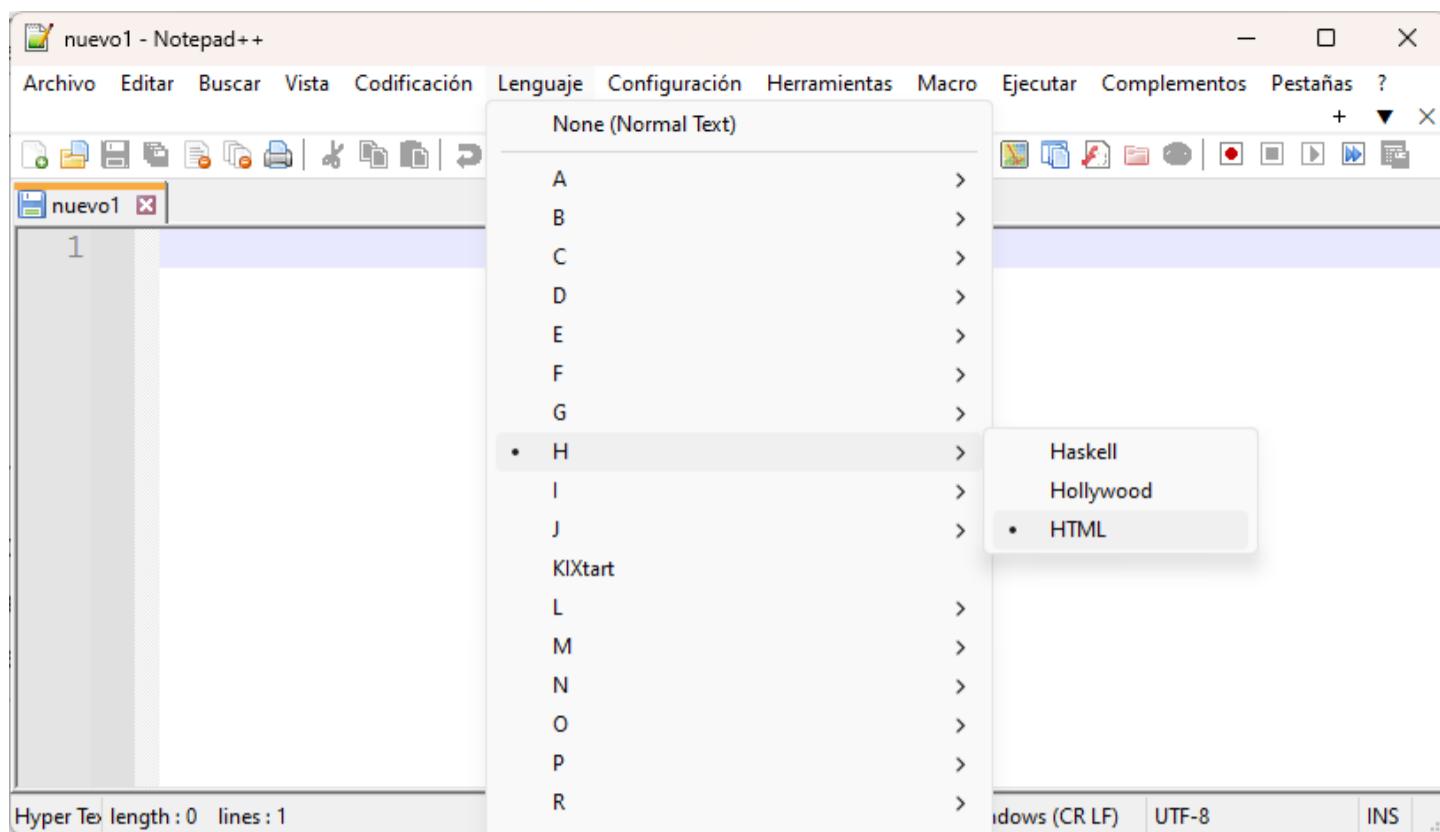


Ilustración 4: Configuramos Notepad++ para editar archivos HTML y JavaScript

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

Escribimos 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 podamos hacer mantenimiento y software que hay que desechar. Debemos usarlos bastante y bien. Similar a C++, si el comentario es de una sola línea, se usa // pero si se requieren varias líneas se puede encerrar entre /* y */ . Ver el código:

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”

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

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

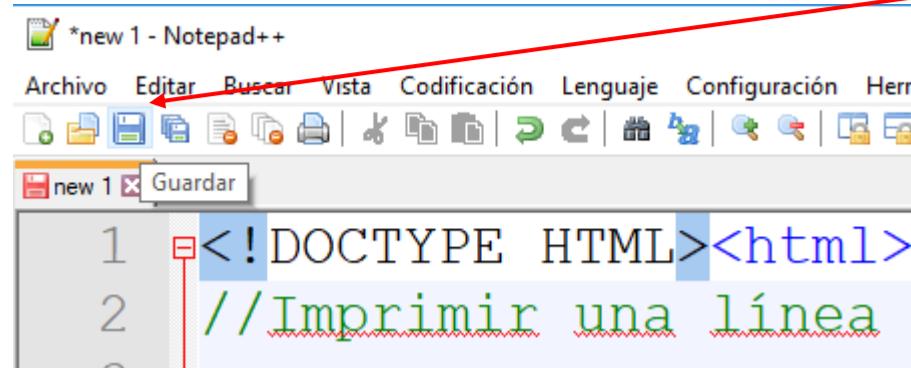


Ilustración 5: Botón de guardar

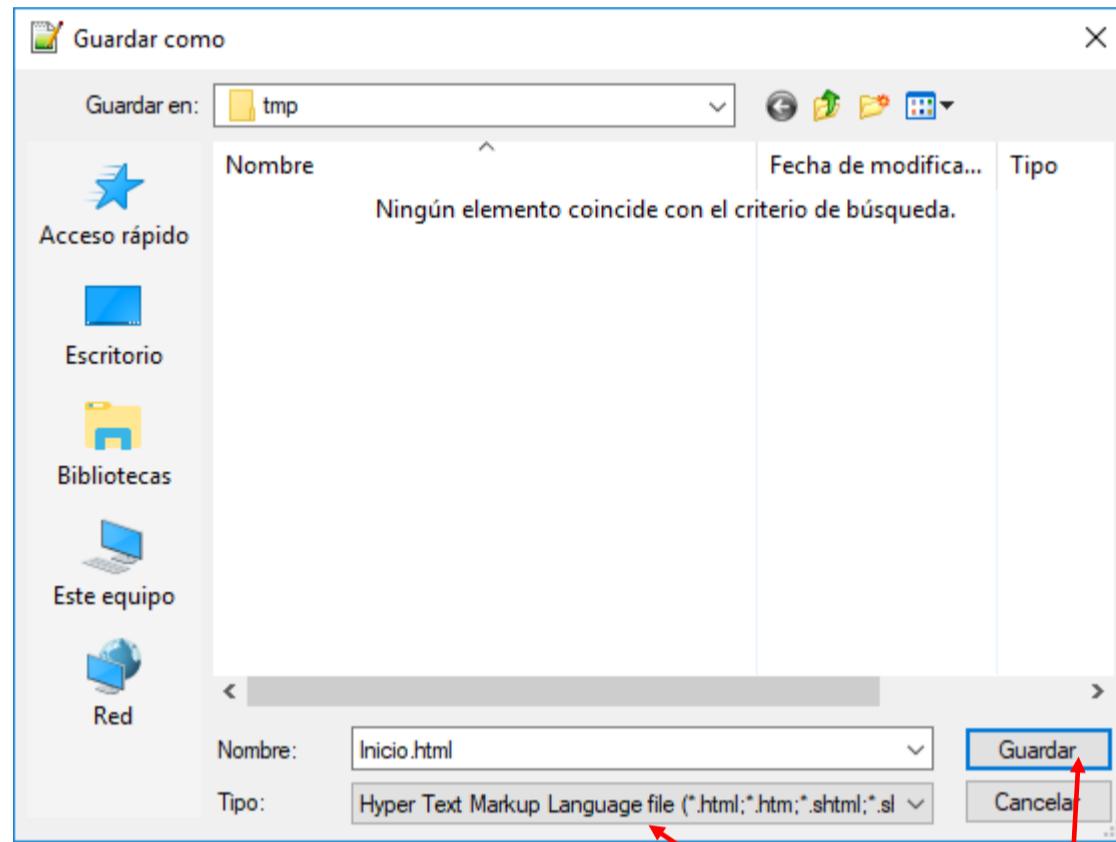


Ilustración 6: Guardar la página HTML en el PC

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

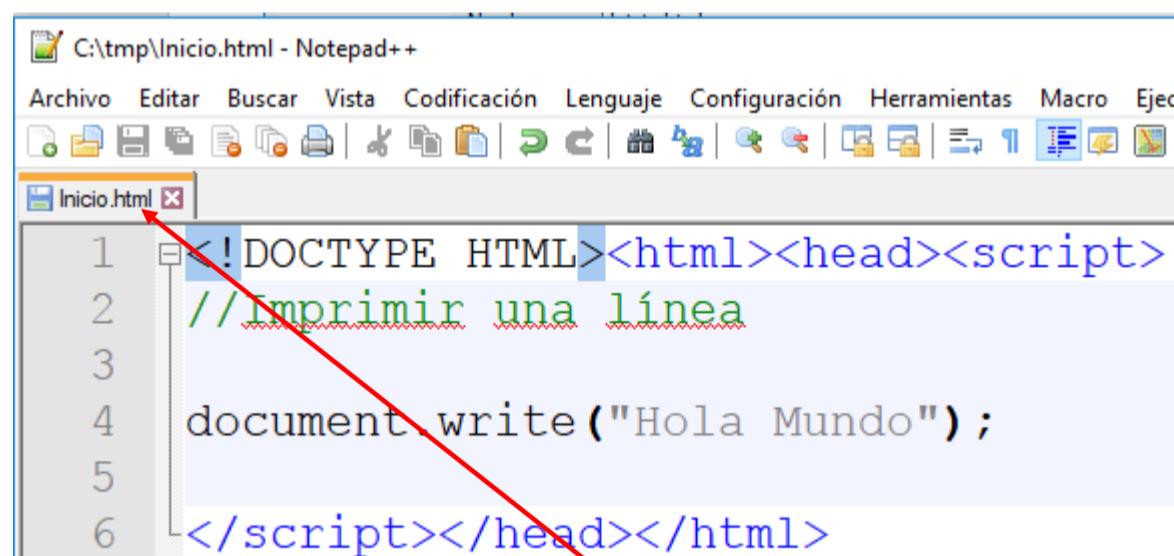


Ilustración 7: Debemos fijarnos que el archivo tenga extensión .html

Hay que fijarnos que la extensión del archivo sea .html

El siguiente paso es ejecutar nuestra página Web, para ello es ir al directorio o carpeta y dar doble clic en el archivo que será ejecutado por el navegador de internet que tengamos por defecto.

003.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>
```

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 podemos 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>
```

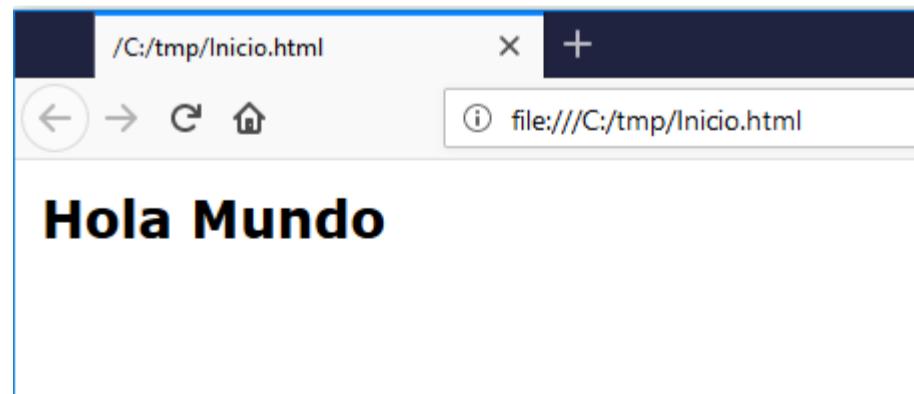


Ilustración 8: Mensaje en negrillas

Uso de variables

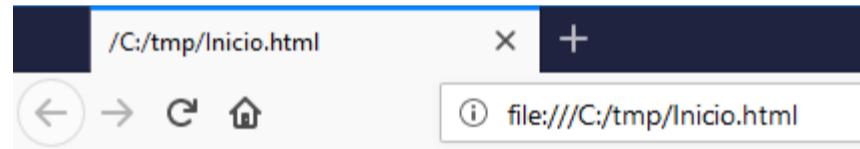
JavaScript declara las variables con la palabra reservada let. Es un lenguaje débilmente tipado, es decir, no definimos variables enteras, reales o de cadena. JavaScript detecta el tipo de dato cuando se le asigna el primer valor a la variable. Si se le asigna un texto, se considera que es una variable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

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.



15
Esta es una prueba
-7.126

Ilustración 9: Impresión del valor de cada variable

Podemos 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;

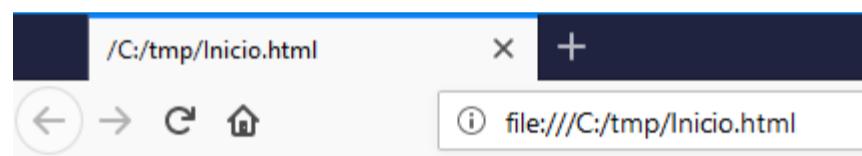
    document.write(valA + "<br>");
    document.write(cadena + "<br>");
    document.write(aproxima);
</script>
</body>
</html>
```



Ilustración 10: Impresión de valor de cada variable

```
<!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>
```



-901
Manual de JavaScript
864.5

Ilustración 11: Impresión de valor de cada variable

Operaciones matemáticas

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

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>
```

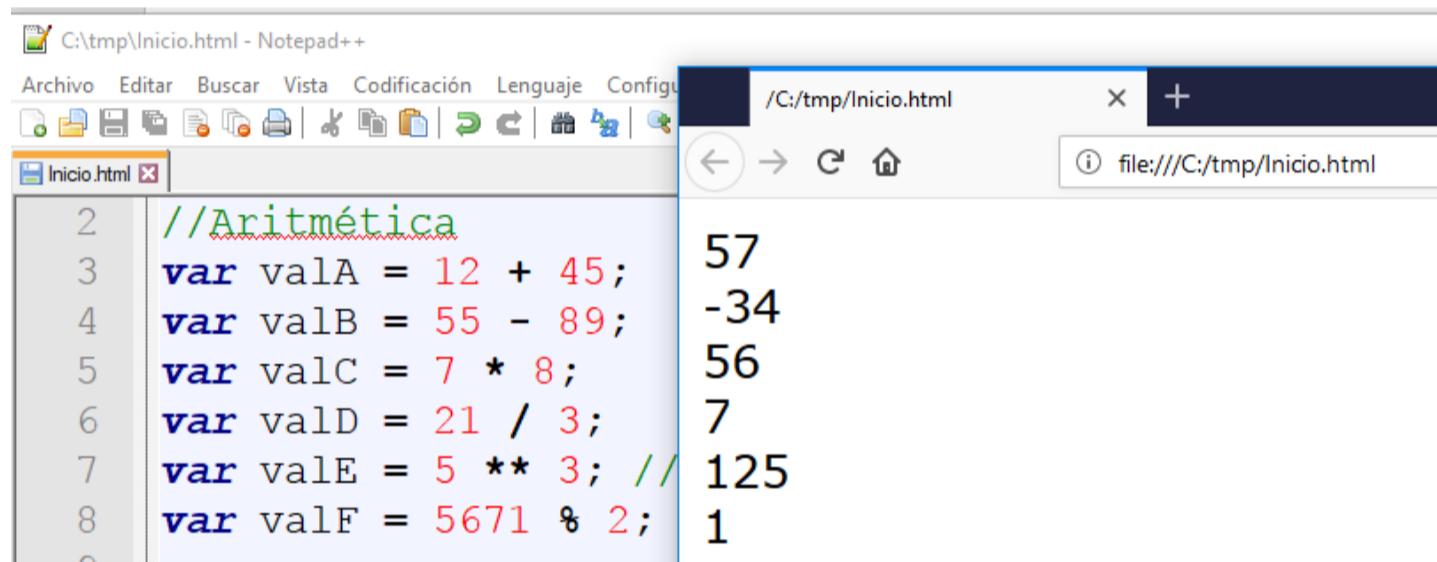


Ilustración 12: Resultado de la ejecución de las operaciones matemáticas

Asignación

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

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>
```

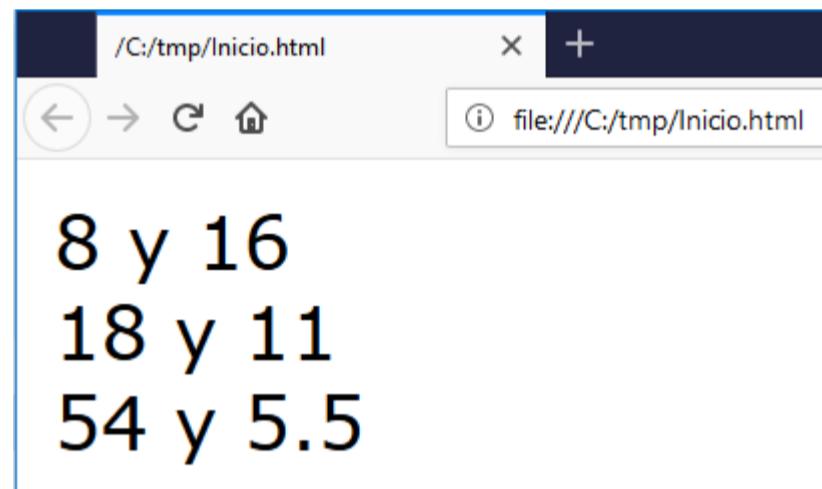


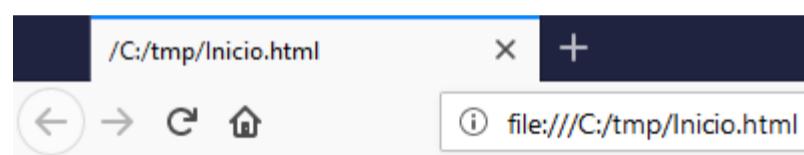
Ilustración 13: Combinando texto con valores

Operaciones con cadenas

Se utiliza el operador + para concatenar cadenas.

011.html

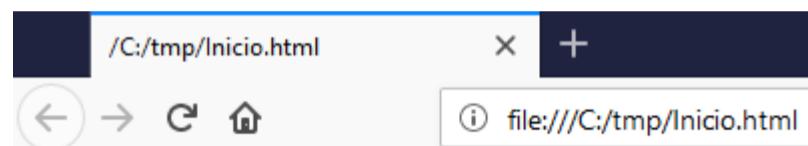
```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Uso de cadenas
    let txtA = "Una";
    let txtB = "Cadena";
    let txtC = txtA + " --- " + txtB;
    document.write(txtC);
</script>
</body>
</html>
```



Unidad 1 - Introducción a JavaScript

Ilustración 14: Resultado de la concatenación

```
<!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>
```



Una --- 45.98

Ilustración 15: Se concatena texto con números

```
<!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>
```

Esto sucede:

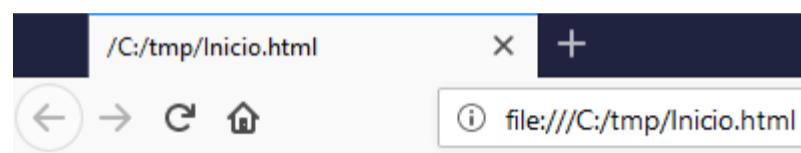


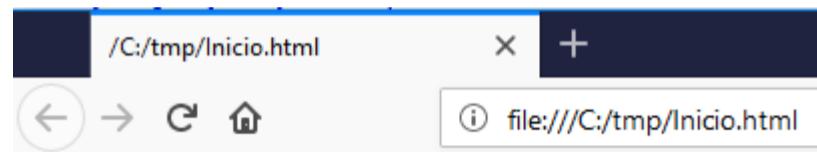
Ilustración 16: Se concatena números y texto

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

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:



12aaaaa

Ilustración 17: Primero se hace la operación matemática y luego la concatenación

```
<!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>
```

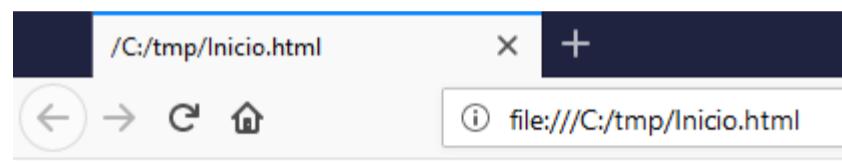


Ilustración 18: Se hace la operación matemática y el resto es concatenación

Impresión de caracteres propios del idioma español

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

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("áéíóúÁÉÍÓÚÑñäëöü¿¡!\"");
4 </script></head></html>
5
```

Ilustración 19: Imprimir caracteres propios del idioma español

Y al ejecutarlo esto sucede en Google Chrome:

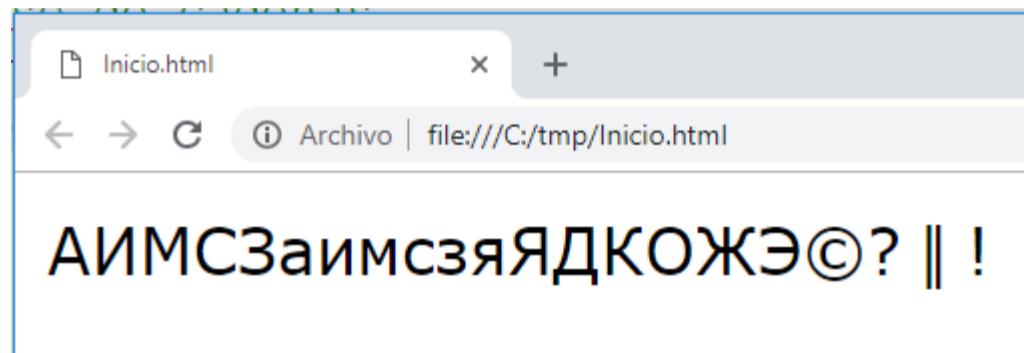


Ilustración 20: En Chrome puede pasar que no se impriman

Funciona correcto en Mozilla Firefox:

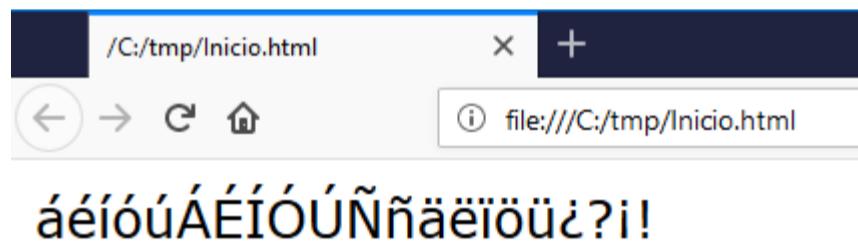


Ilustración 21: Funciona en Mozilla Firefox

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

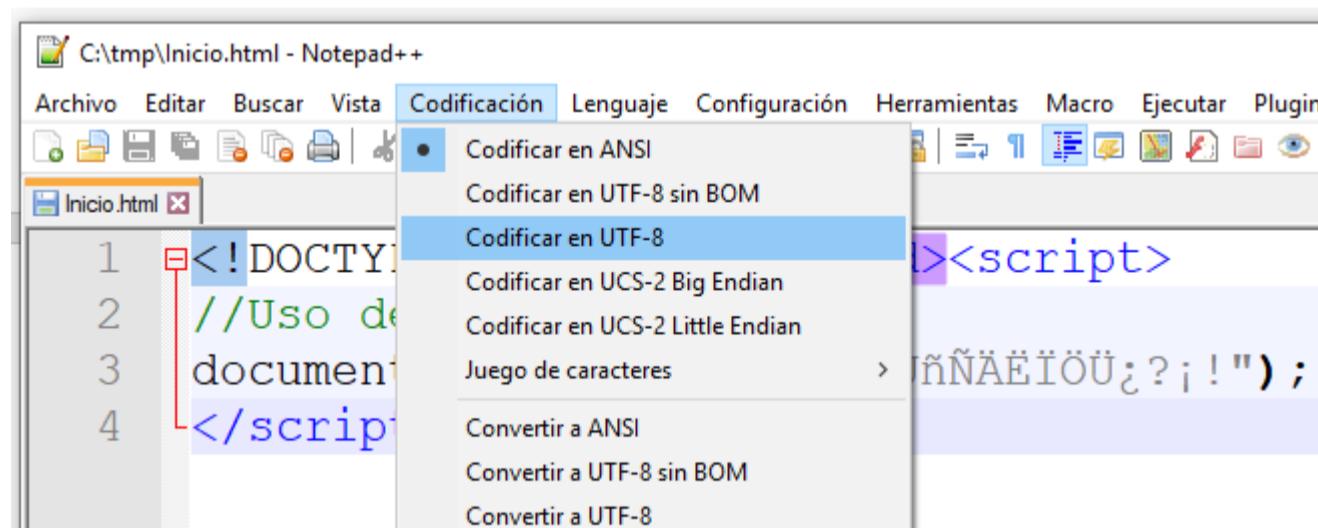


Ilustración 22: Ante un fallo así, verificamos la codificación. Usualmente está en "Codificar en ANSI"

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:

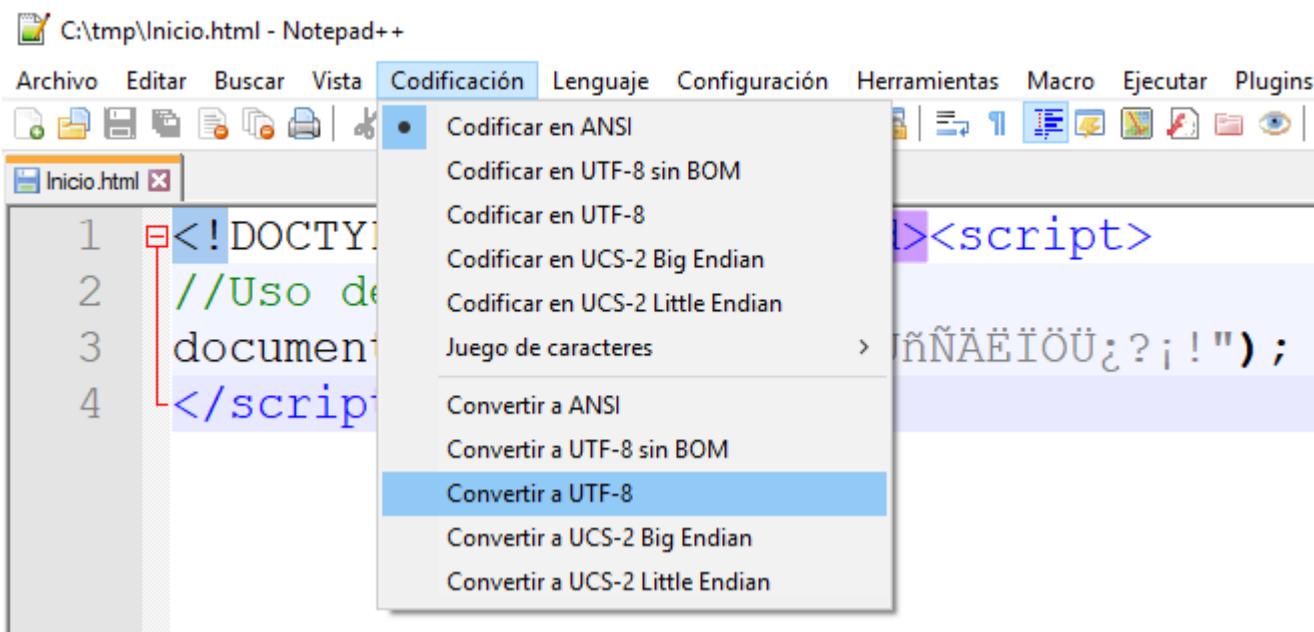


Ilustración 23: Seleccionamos la opción de "Convertir a UTF-8"

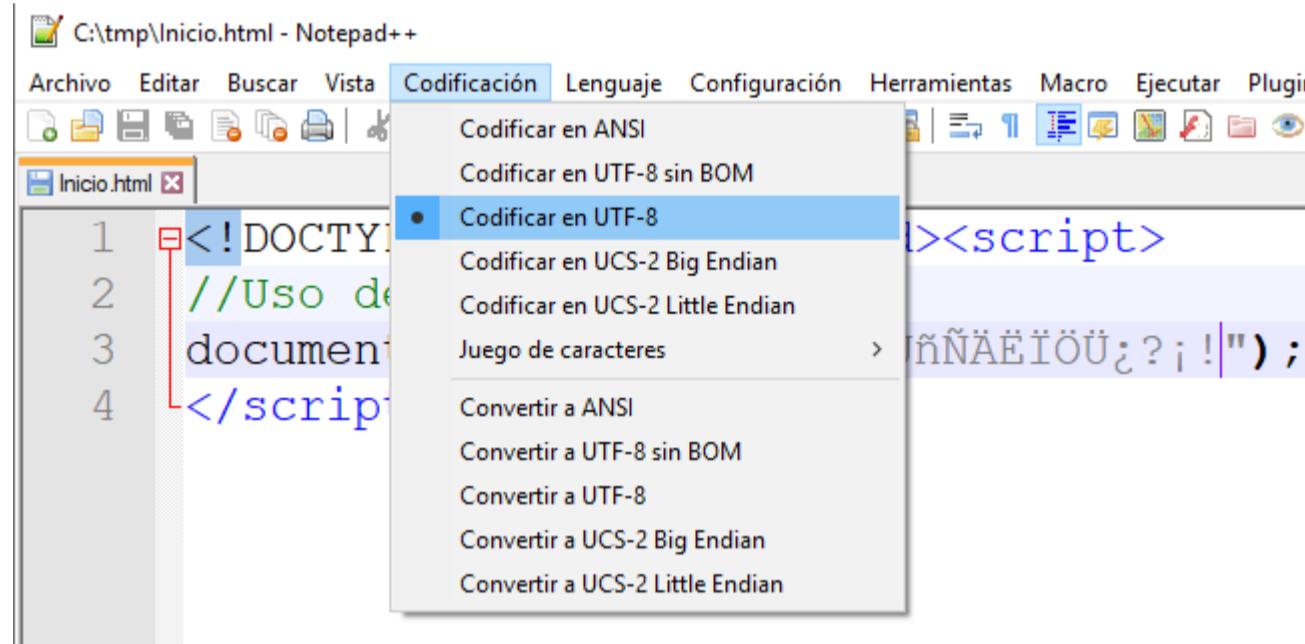
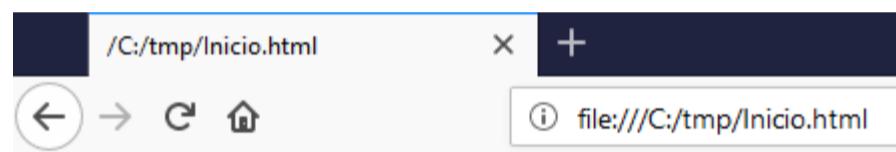


Ilustración 24: Despues de guardar, chequeamos que esté en la opción "Codificar en UTF-8"

```
<!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>
```



2.718281828459045
0.6931471805599453
2.302585092994046
1.4426950408889634
0.4342944819032518
3.141592653589793
0.7071067811865476
1.4142135623730951

Ilustración 25: Constantes matemáticas

Funciones trigonométricas

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

017.html

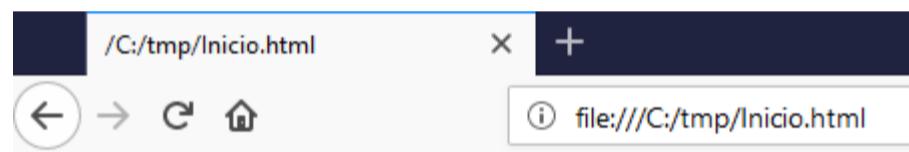
```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Librería matemática.
    let angGrados = 35;
    document.write("Grados: " + angGrados);
    let angRadian = angGrados * Math.PI / 180;
    document.write("<br>Radianes: " + angRadian);
    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>
```



Grados: 35
Radianes: 0.6108652381980153
Seno: 0.573576436351046
Coseno: 0.8191520442889918
Tangente: 0.7002075382097097
Arcoseno: 0.6108652381980153
Arcocoseno: 0.6108652381980153
Arcotangente: 0.6108652381980153

Ilustración 26: Funciones trigonométricas

```
<!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>
```



Raíz cúbica: 11
Techo: 5
Exponencial: 2.7182818284590455
Piso: 4
Log: 0.6931471805599453
Mayor: 9
Menor: 3
Potencia: 32
Aleatorio: 0.26238171616841743
Redondeo: 4
Redondeo: 3
Redondeo: 4
Raíz Cuadrada: 12
Truncar: 8

Ilustración 27: Funciones matemáticas

```
<!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>
```

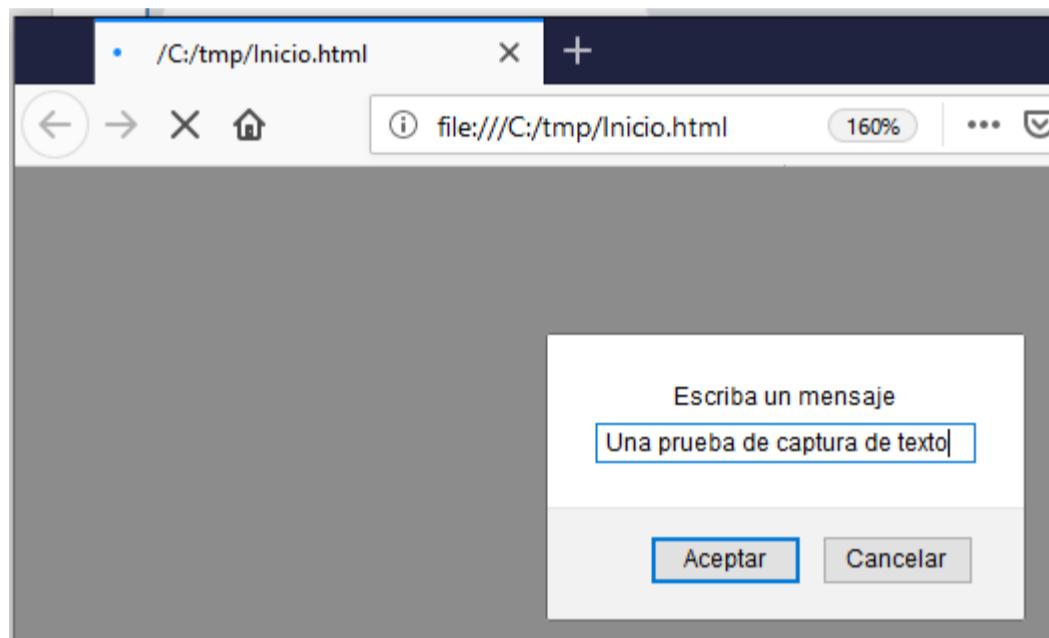


Ilustración 28: Cuadro de diálogo para capturar un valor texto

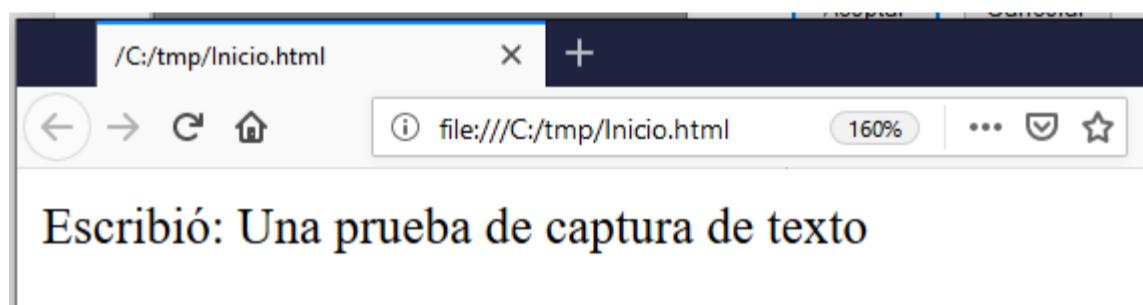


Ilustración 29: Muestra el valor capturado

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>
```

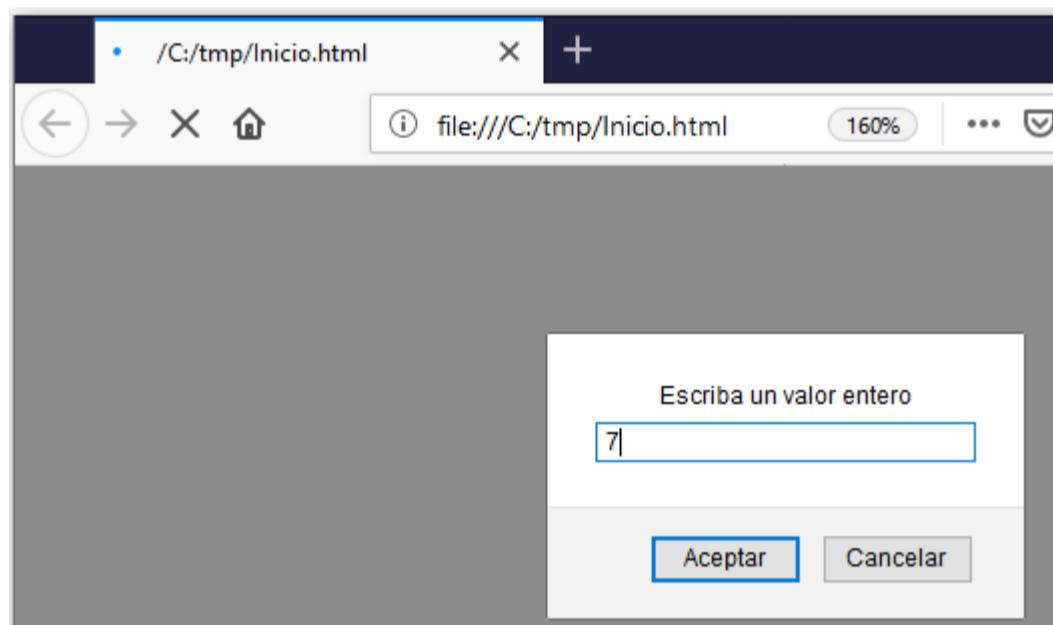


Ilustración 30: Cuadro de diálogo que captura un valor texto

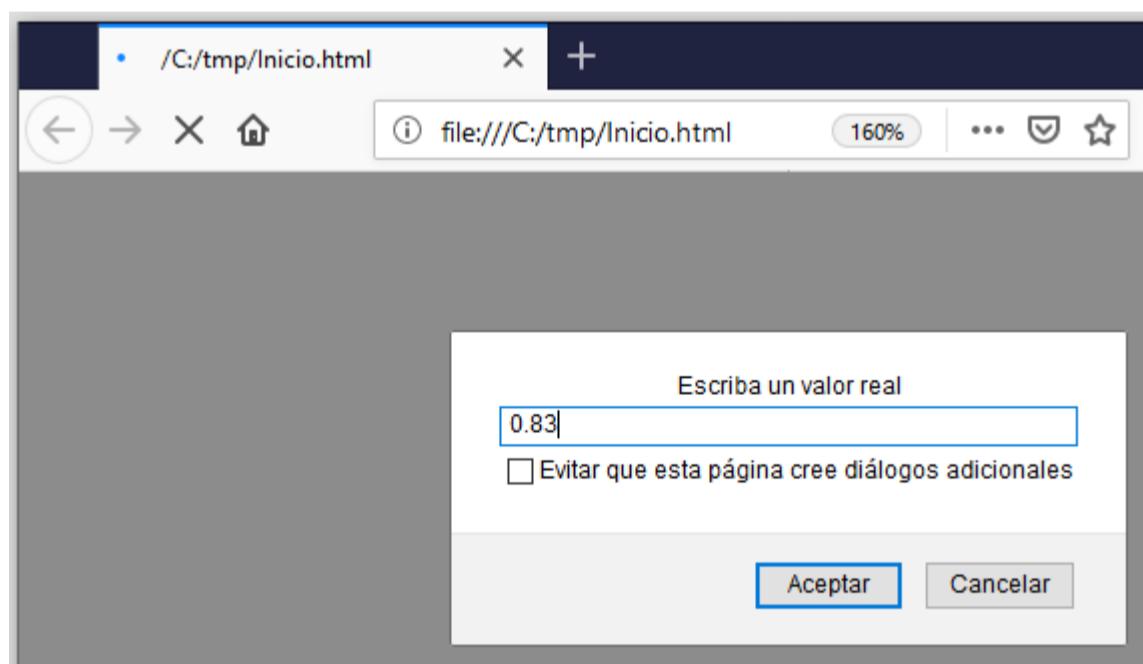


Ilustración 31: Capturando un valor texto que después se convertirá en real

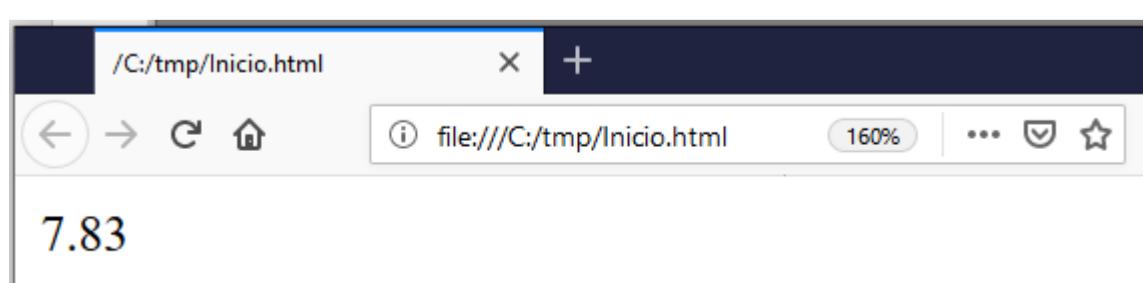


Ilustración 32: Resultado de la operación matemática

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 nos permiten, por ejemplo, detectar errores en nuestro código.

Tenemos el siguiente código con algunos errores:

```
C:\Users\engin\OneDrive\Proyecto\Libro\22. JSAlgoritmo>ErrorSintaxis.html - Notepad++
Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Complementos Pestañas ?
Rafael Alberto Moreno Parra (rafael)

ErrorSintaxis.html x
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(prompt("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 33: Código en JavaScript con varios errores

Al ejecutar en Edge, la pantalla aparece en blanco

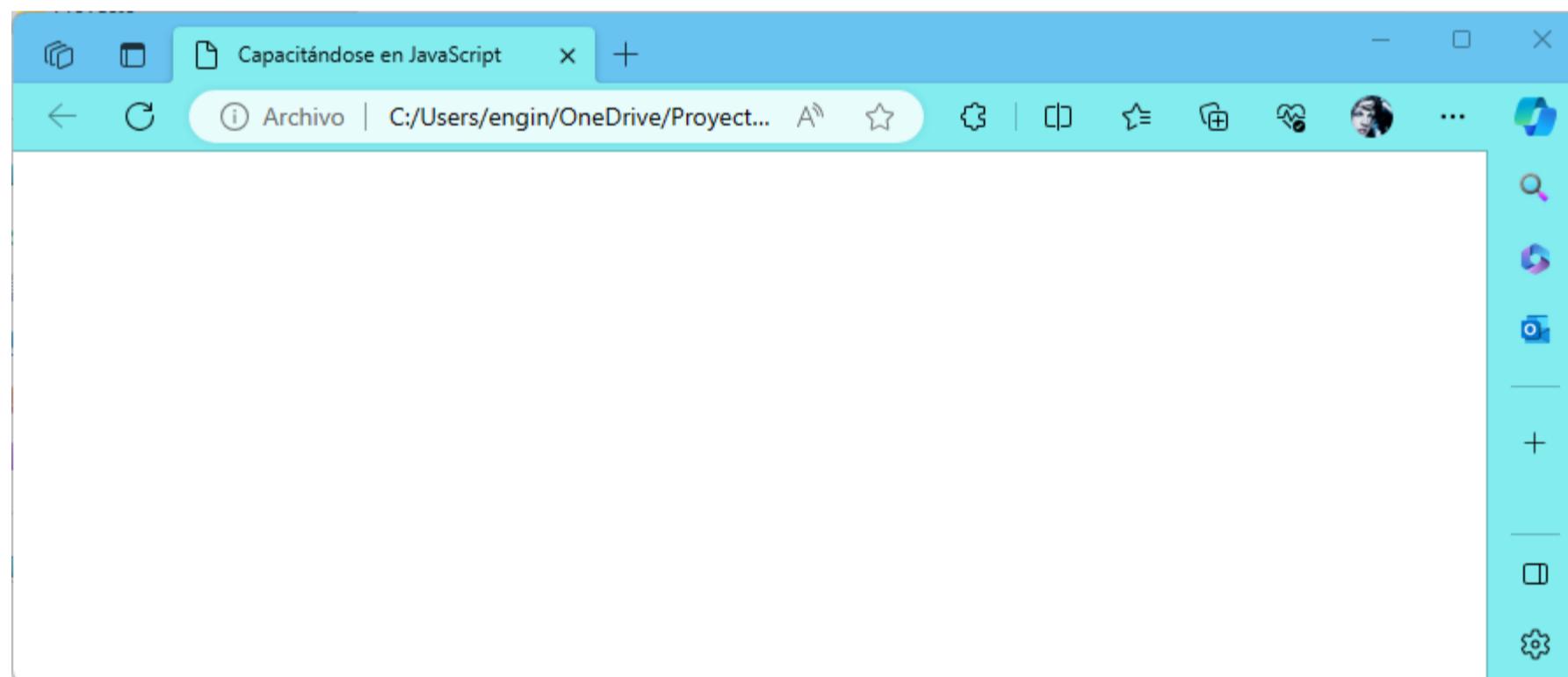


Ilustración 34: Ante errores en JavaScript, el navegador queda en blanco

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

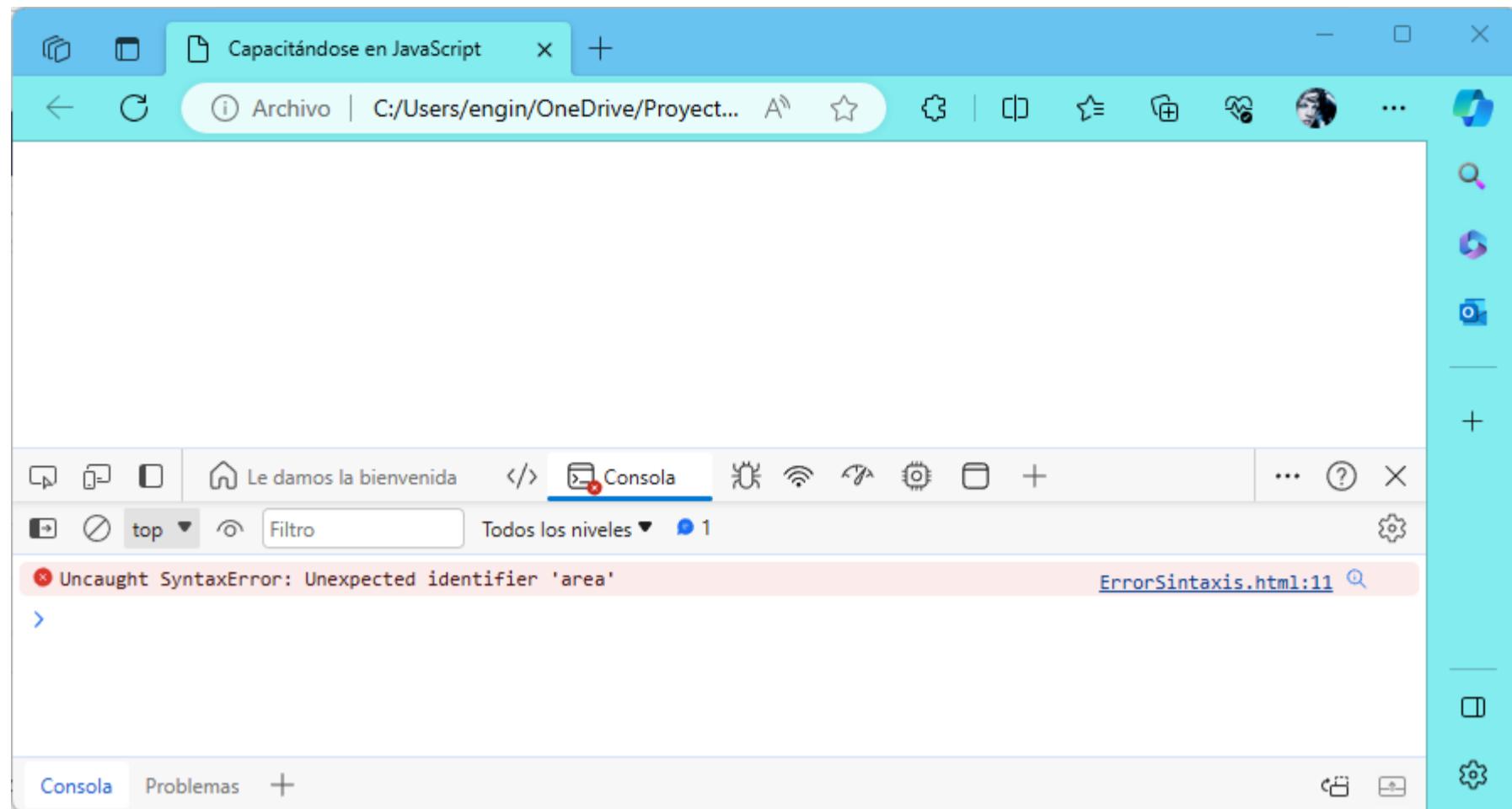


Ilustración 35: Presionando F12 en el navegador Edge y dando clic en Consola se muestra el error

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

Herramienta de Google para programar en JavaScript

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

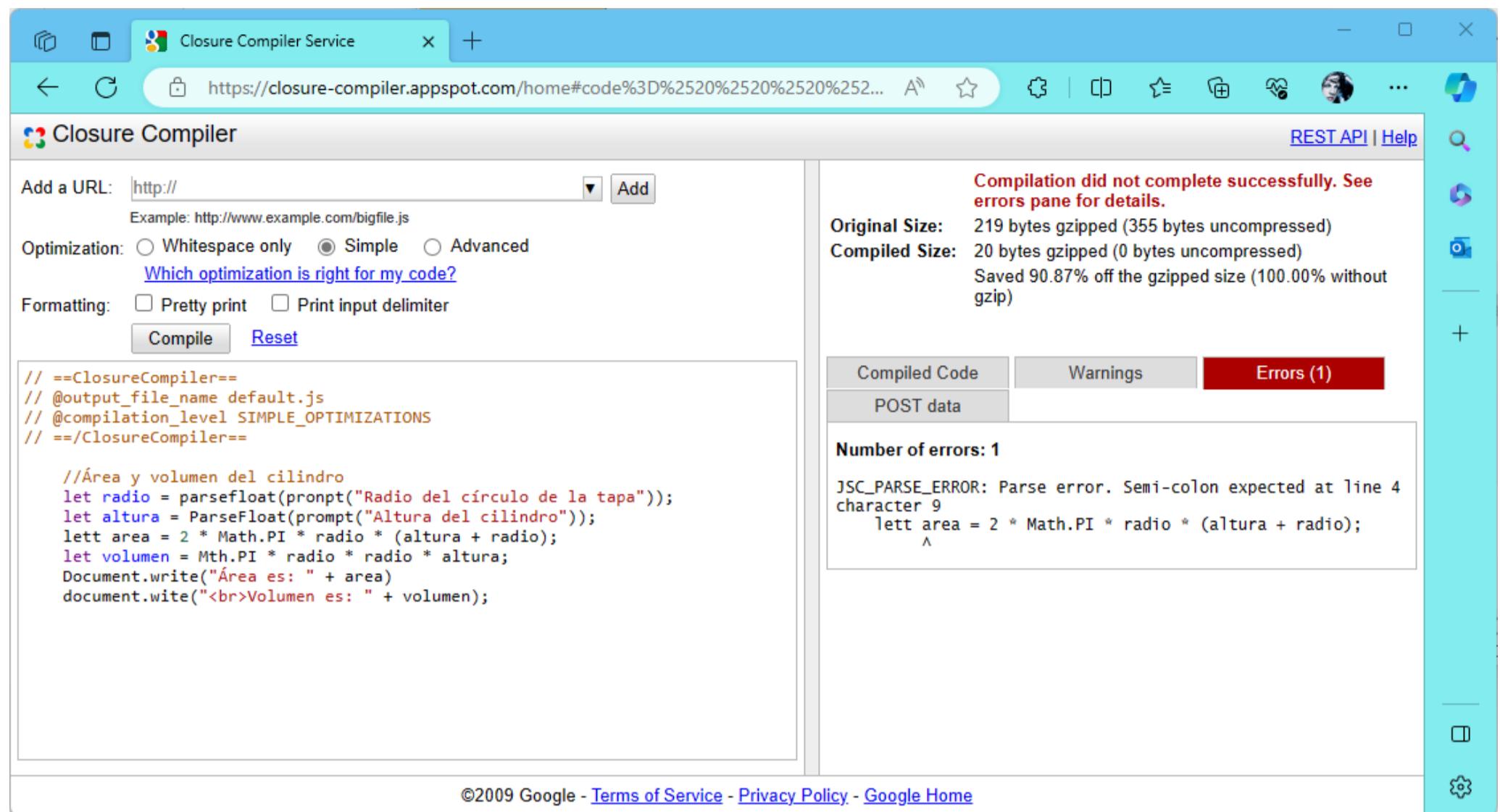


Ilustración 36: Closure Compiler para optimizar código en JavaScript

Validación en línea del código

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

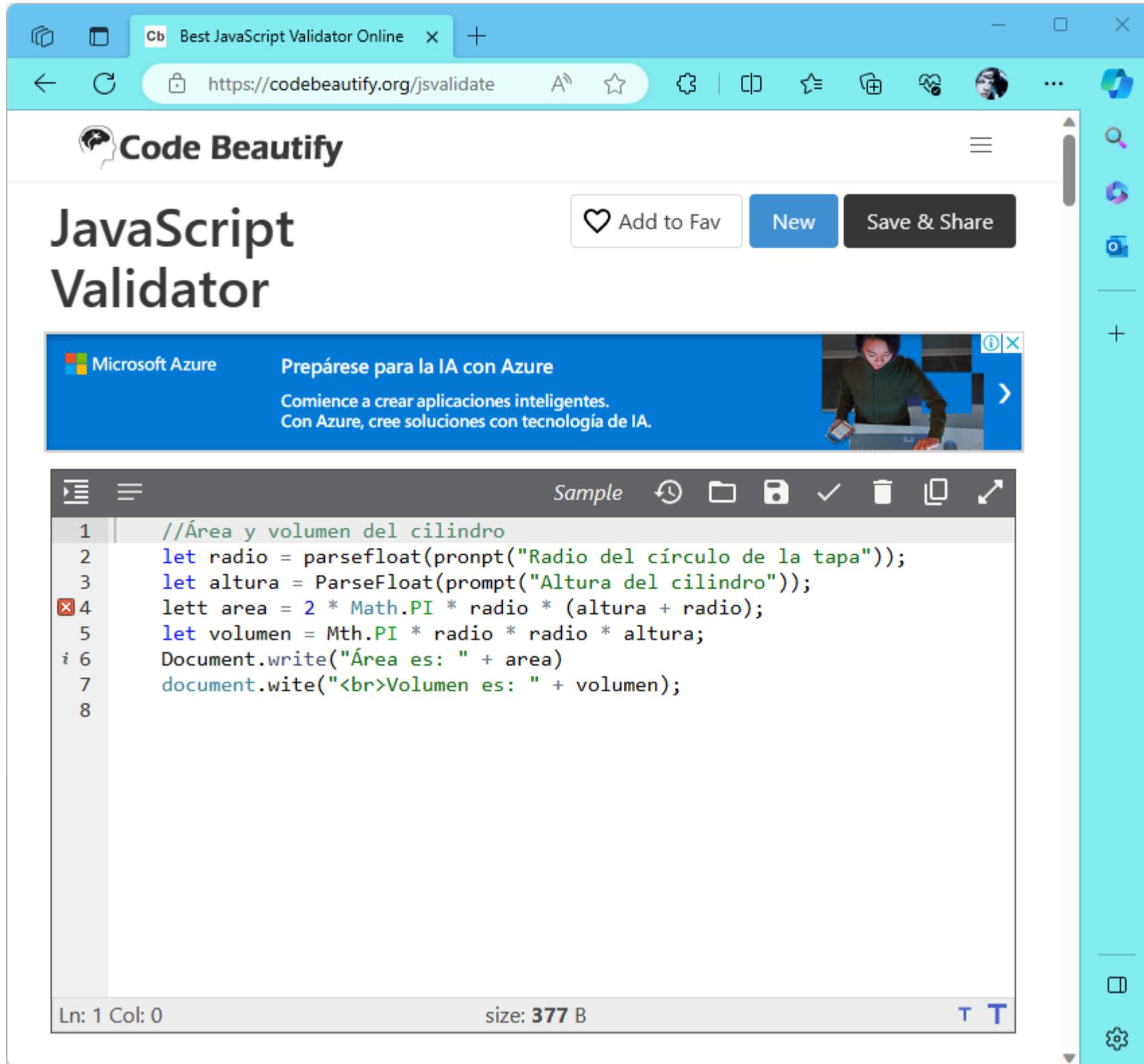


Ilustración 37: JavaScript Validator para chequear sintaxis

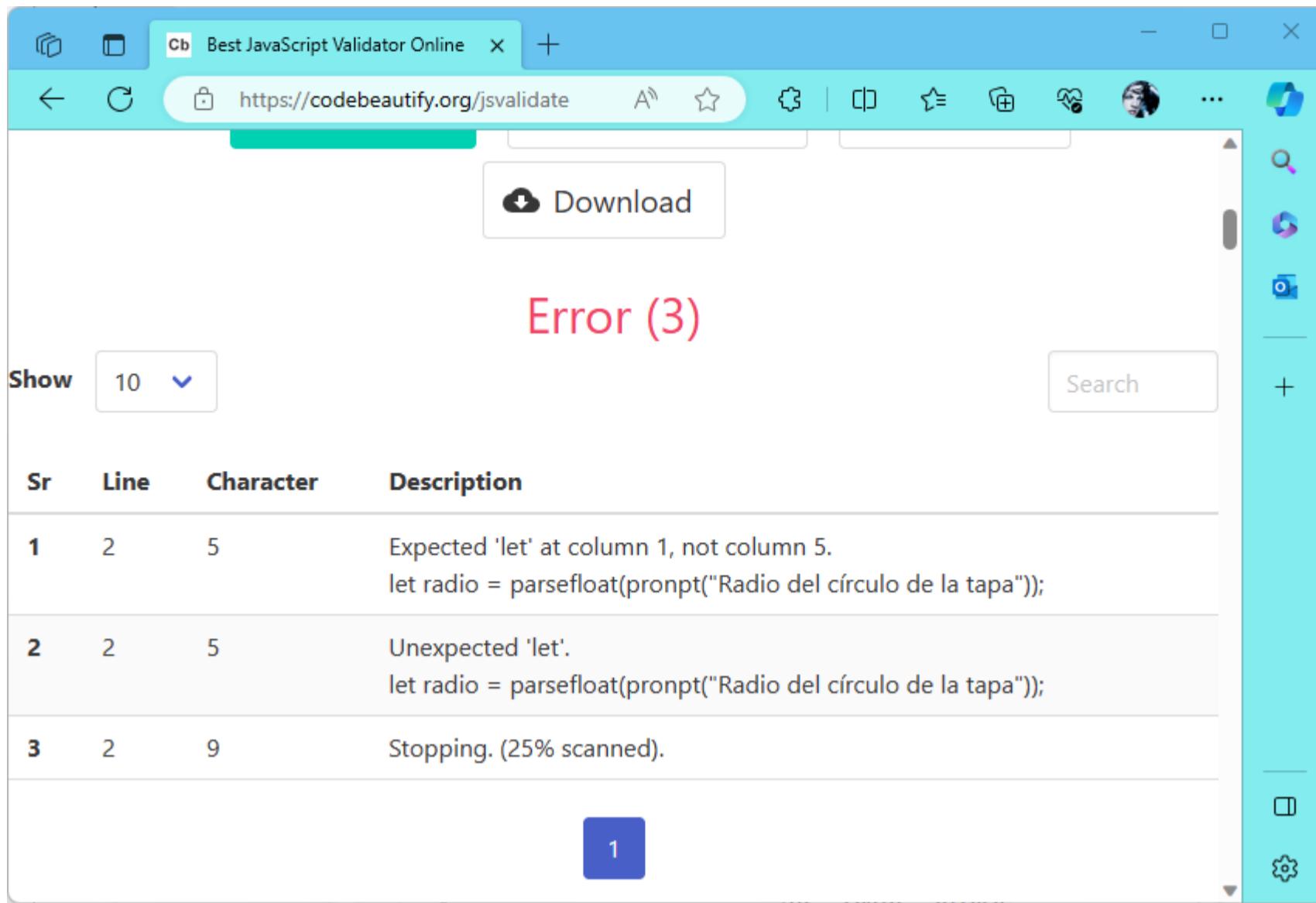


Ilustración 38: Diagnóstico de JavaScript Validator de un script de ejemplo

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

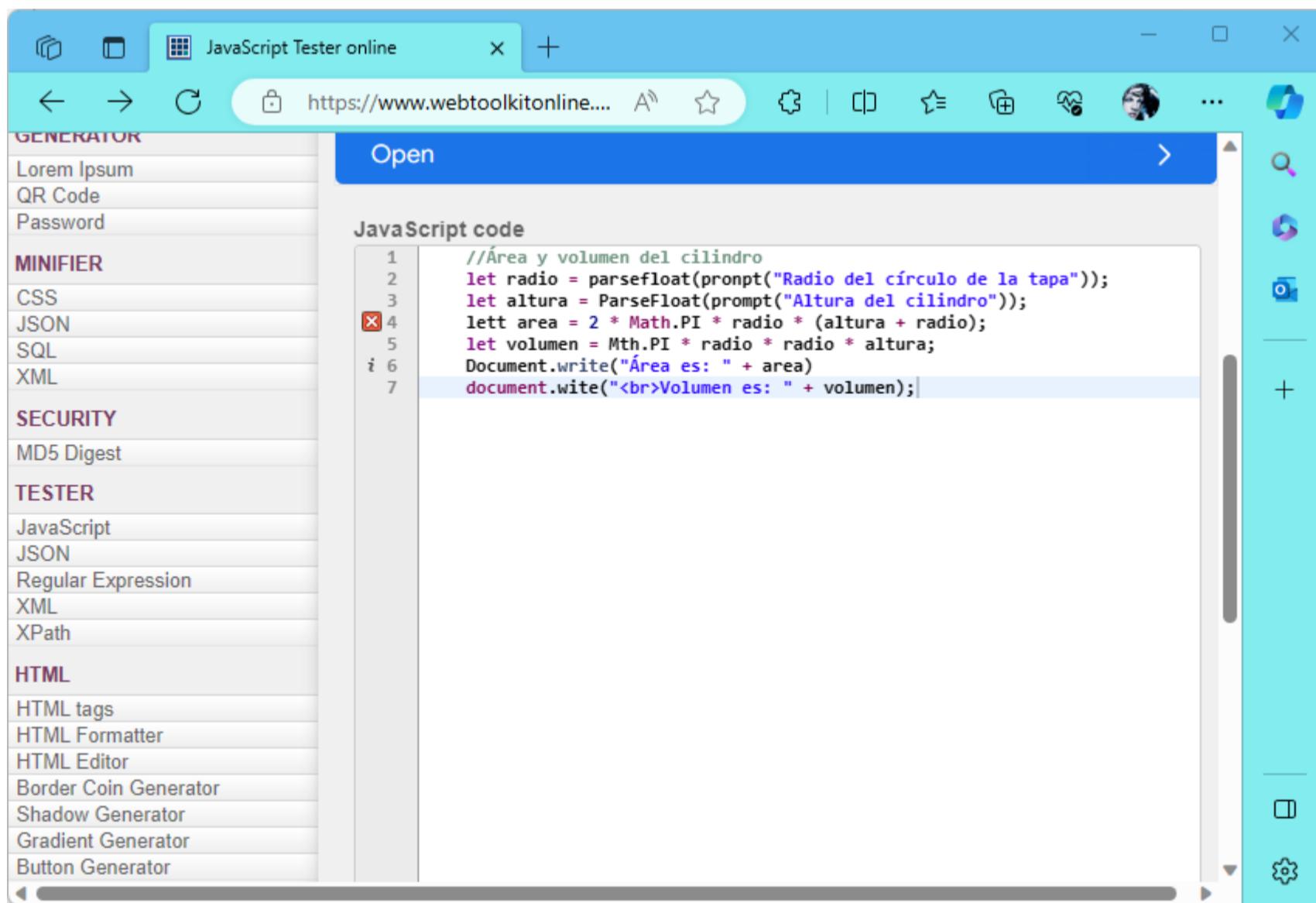


Ilustración 39:JavaScript Tester para validar sintaxis

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

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

```
//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 = Mth.PI * radio * radio * altura;
Document.write("Área es: " + area)
document.wite("<br>Volumen es: " + volumen);
```

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 right sidebar features the **JSHint** logo and version 2.13.6, with links to About, Documentation, Install, Contribute, and Blog. A vertical toolbar on the far right includes icons for search, refresh, and settings.

Ilustración 40: 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>
```

Sé 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>
```

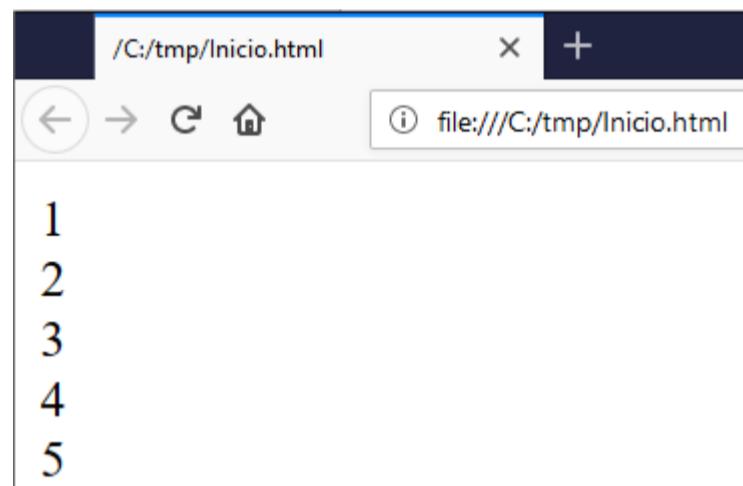


Ilustración 41: Ejecución del ciclo ascendente

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>
```

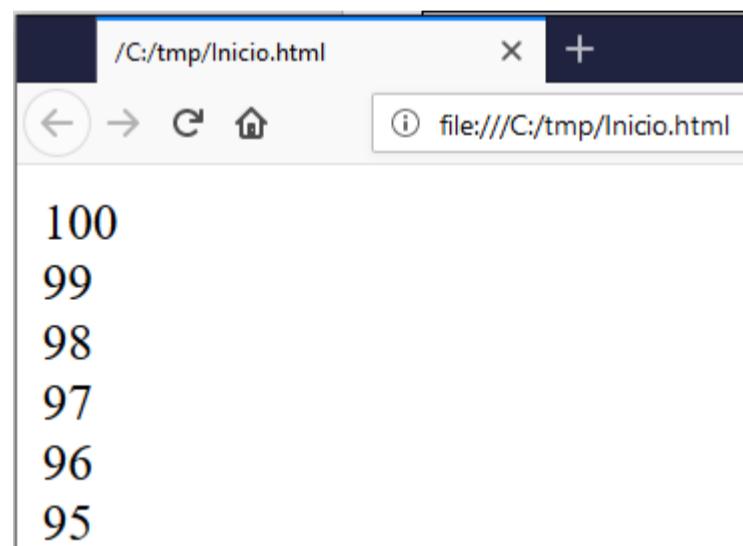


Ilustración 42: Ejecución del ciclo descendente

```
<!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>
```

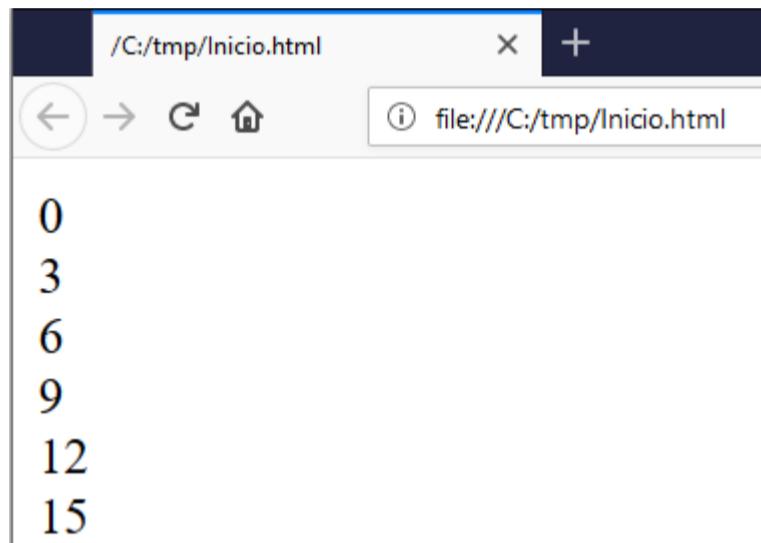


Ilustración 43: Ciclo ascendente de 3 en 3

```
<!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>
```

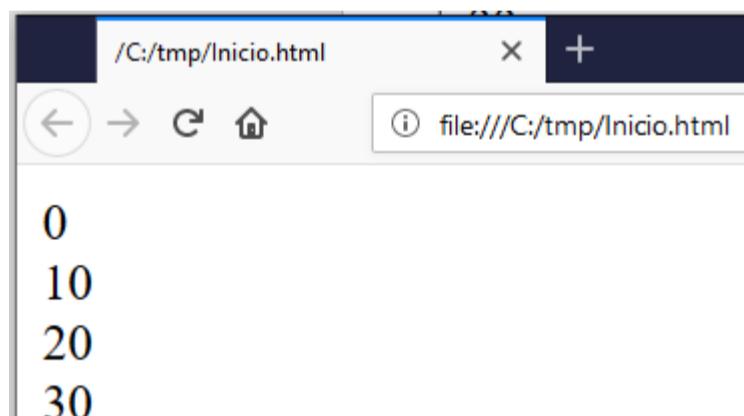


Ilustración 44: Ciclo ascendente de 10 en 10

```
<!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>
```



Ilustración 45: Ciclo ascendente doblando el anterior

```
<!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>
```

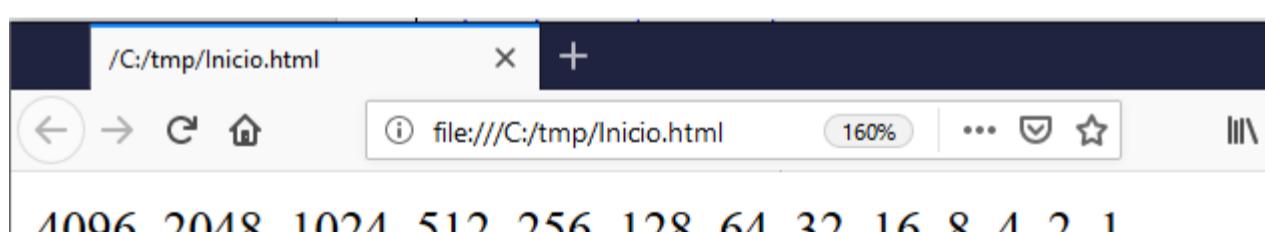
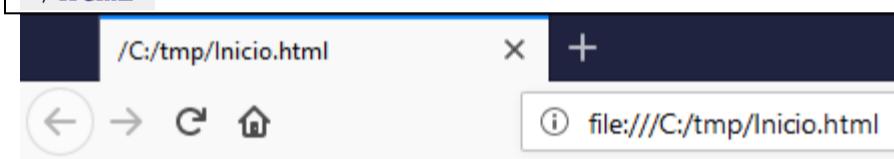


Ilustración 46: Ciclo descendente dividiendo entre 2 el anterior

```
<!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>
```



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

Ilustración 47: Ciclo que muestra el progreso de dos variables

```
<!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>
```

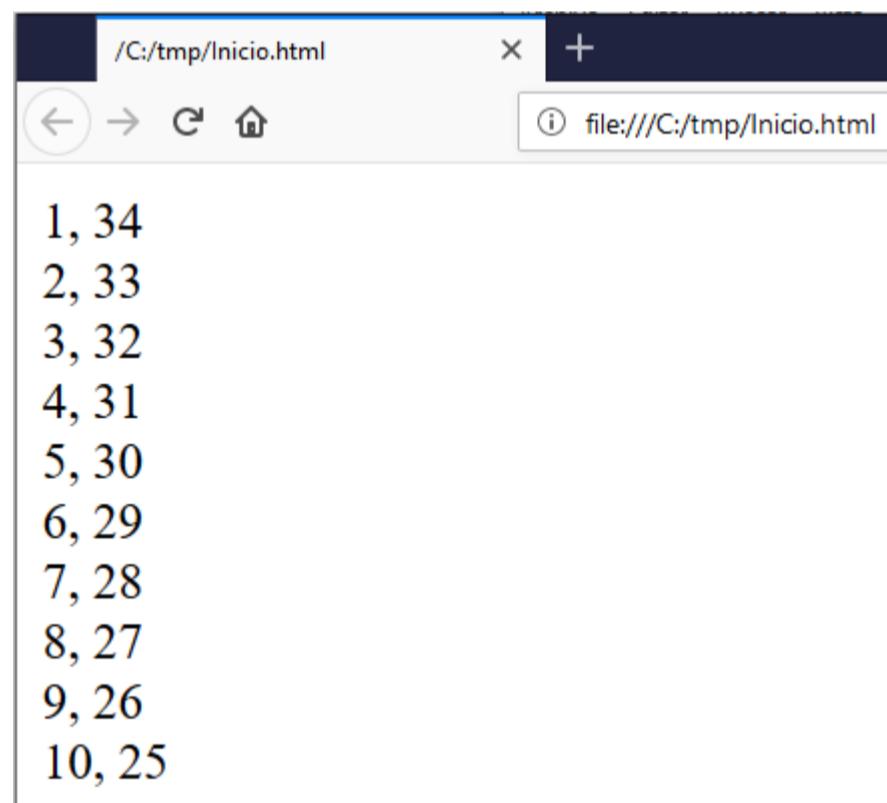


Ilustración 48: Ciclo que muestra el progreso de dos variables:

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>
```

54

```
<!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>
```

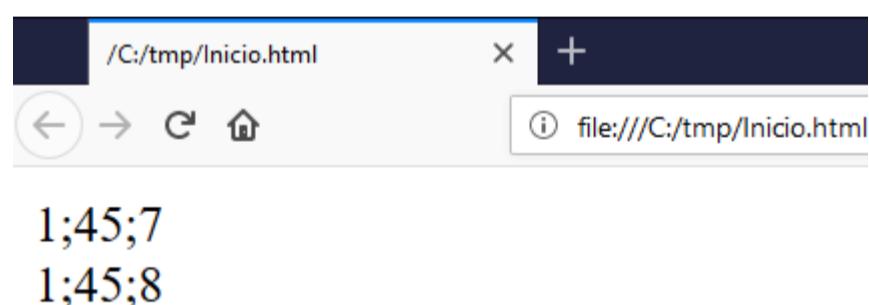


Ilustración 49: Ciclos anidados y salir de estos

```
<!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>
```



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

Ilustración 50: Ciclos anidados y salir de estos

```
<!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

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

Ilustración 51: Funciones, subrutinas o procedimientos

```

//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;
}

```

```
<!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>
```



Ilustración 52: Entre 1000 y 9999. Cantidad de números que la suma de las tres últimas cifras es par: 4500

```

<!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>

```

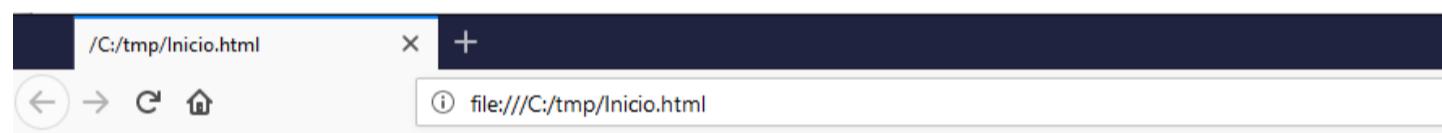


Ilustración 53: Entre 1000 y 9999. Cantidad de números que la multiplicación de las tres últimas cifras es igual a la suma de esas tres últimas cifras: 63

```

<!DOCTYPE HTML>
<html 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>

```

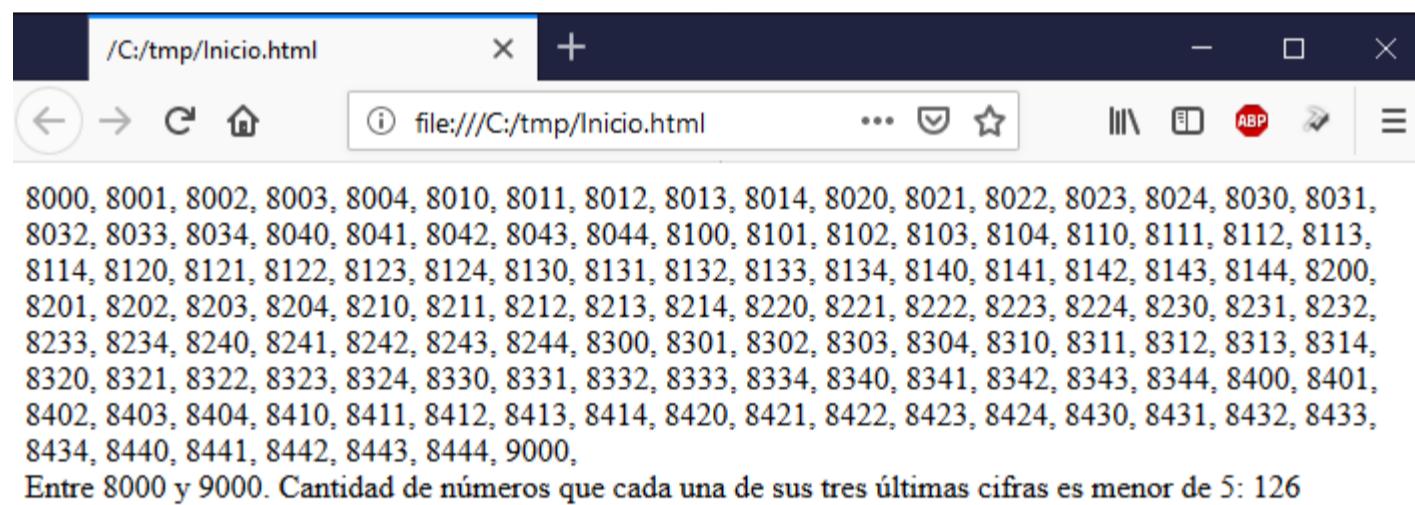


Ilustración 54: Entre 8000 y 9000. Cantidad de números que cada una de sus tres últimas cifras es menor de 5: 126

```

<!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>

```

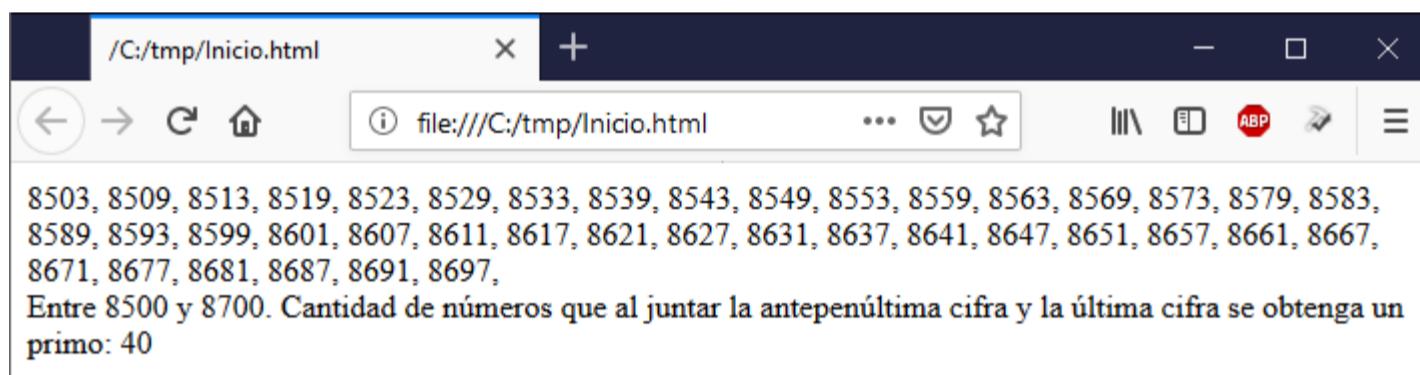


Ilustración 55: Entre 8500 y 8700. Cantidad de números que al juntar la antepenúltima cifra y la última cifra se obtenga un primo: 40

```

<!DOCTYPE HTML>
<html 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>

```

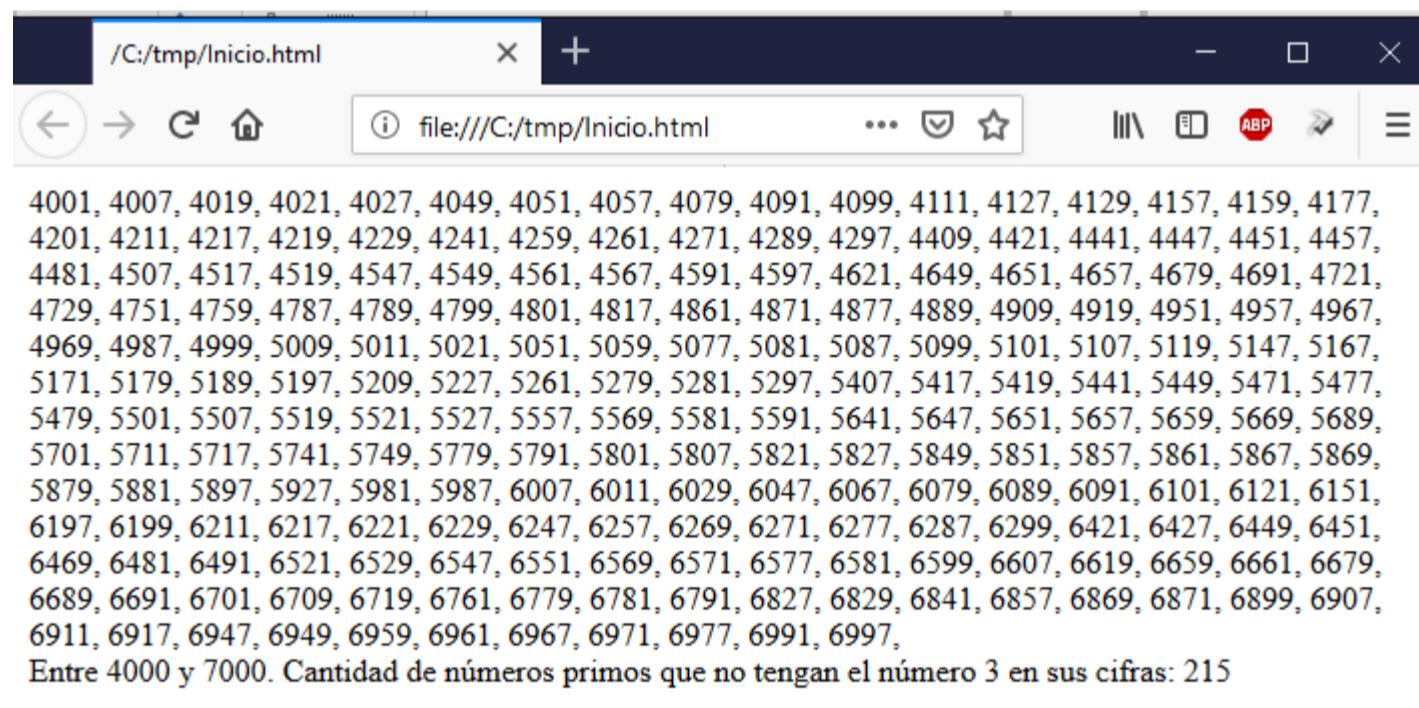


Ilustración 56: Entre 4000 y 7000. Cantidad de números primos que no tengan el número 3 en sus cifras: 215

```

<!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>

```



Ilustración 57: Entre 1000 y 9999. Cantidad de números palíndromos que tengan mínimo dos veces el 7 en sus cifras: 18

```

<!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>

```

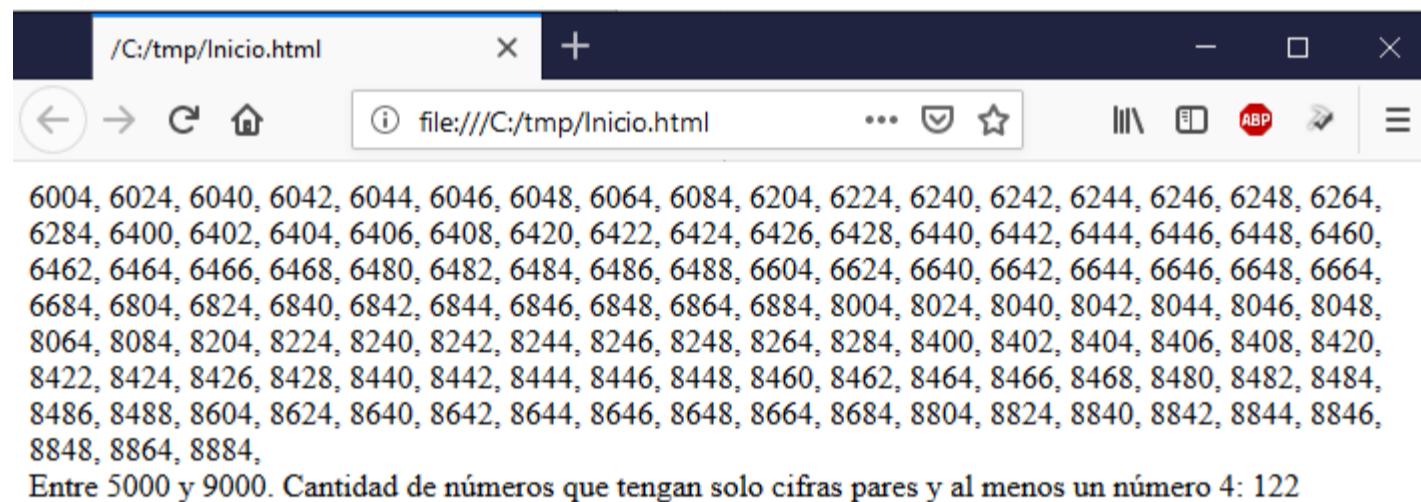


Ilustración 58: Entre 5000 y 9000. Cantidad de números que tengan solo cifras pares y al menos un número 4: 122

```

<!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>

```

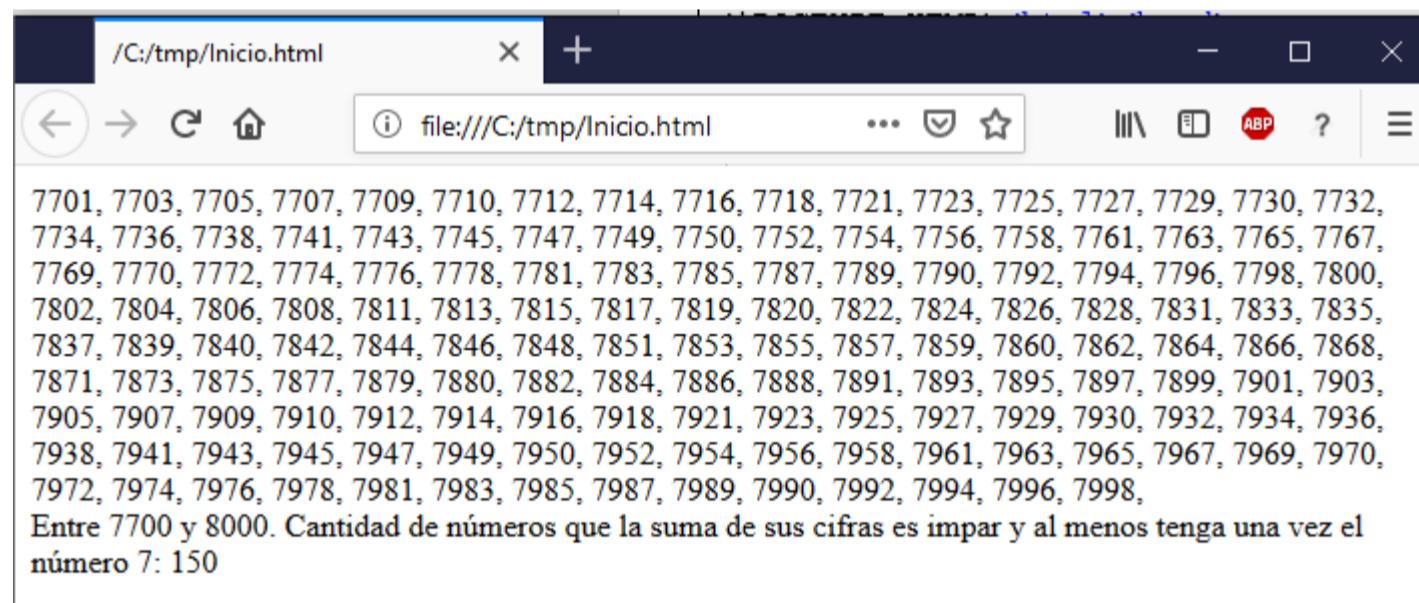


Ilustración 59: Entre 7700 y 8000. Cantidad de números que la suma de sus cifras es impar y al menos tenga una vez el número 7: 150

```

<!DOCTYPE HTML>
<html 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>

```

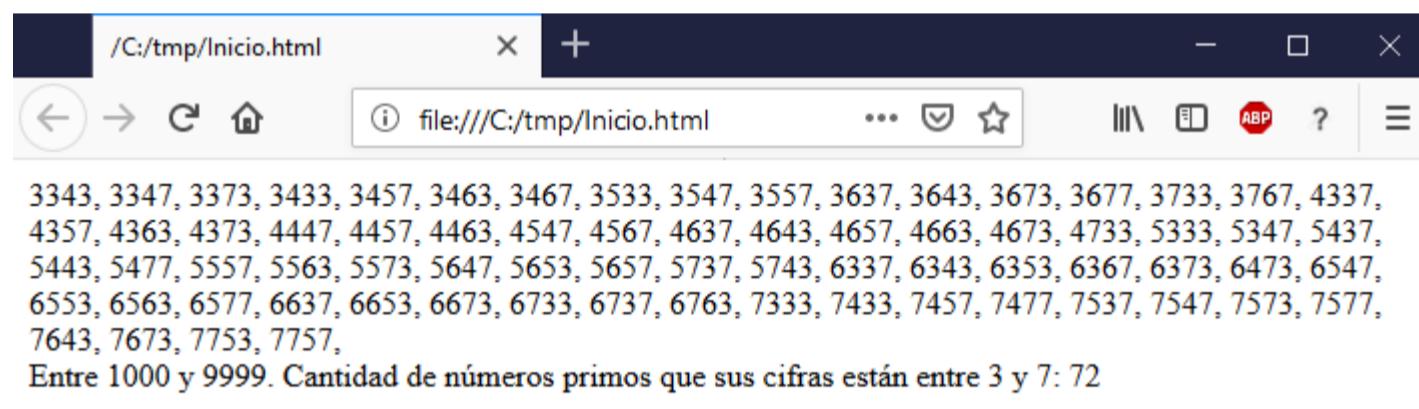


Ilustración 60: Entre 1000 y 9999. Cantidad de números primos que sus cifras están entre 3 y 7: 72

```

<!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>

```

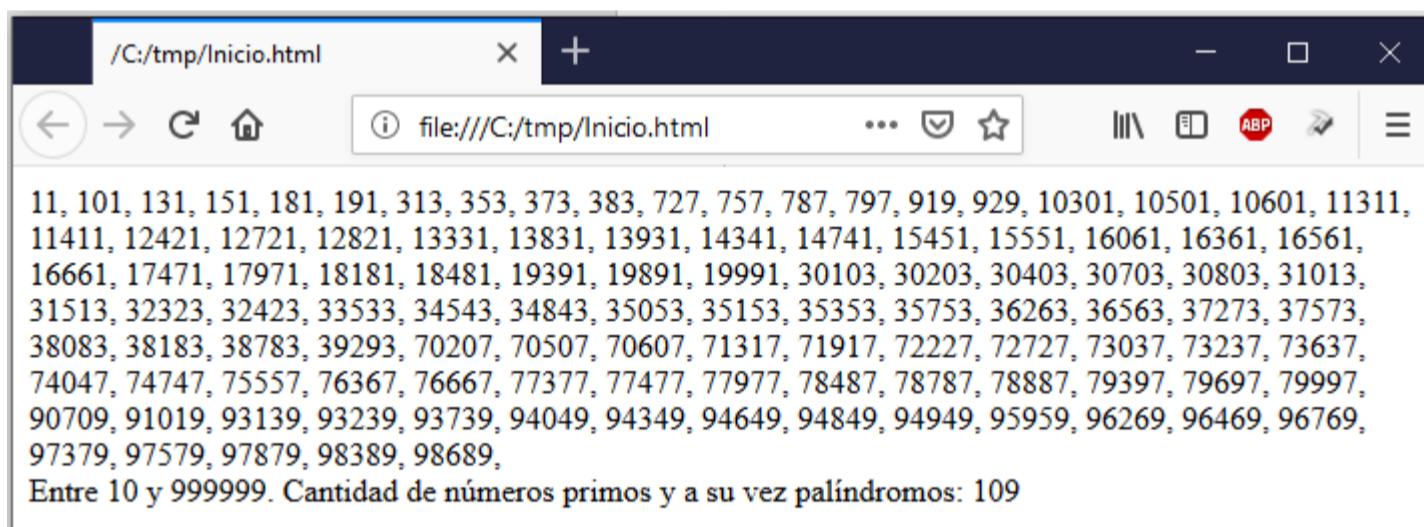


Ilustración 61: Entre 10 y 999999. Cantidad de números primos y a su vez palíndromos: 109

Método de Bisección

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

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>
```

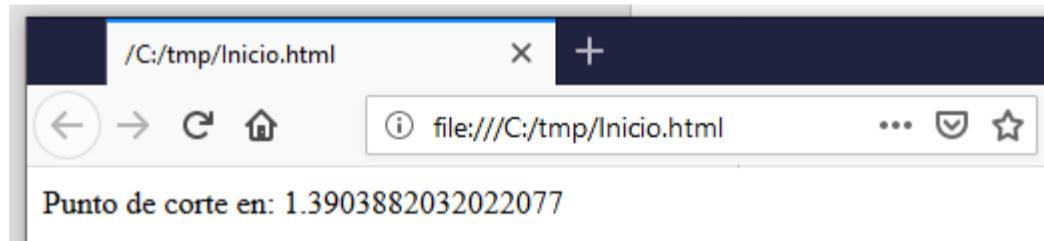


Ilustración 62: Método de bisección

```
<!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>
```

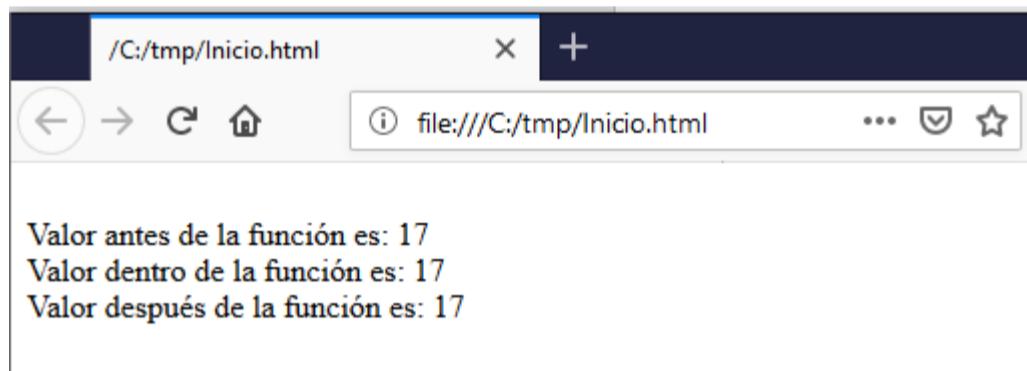


Ilustración 63: Ámbito (scope) de una variable

```

<!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

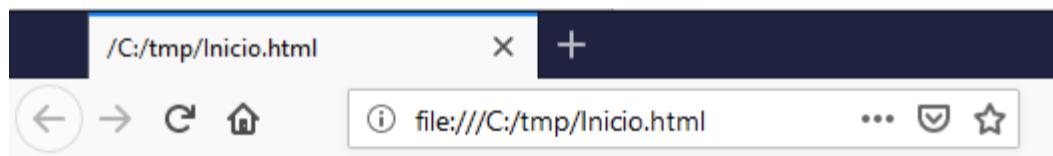
    //Imprime 8906
    document.write("<br>Valor después de la función es: " + valor);

    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>

```



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

Ilustración 64: 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>

```

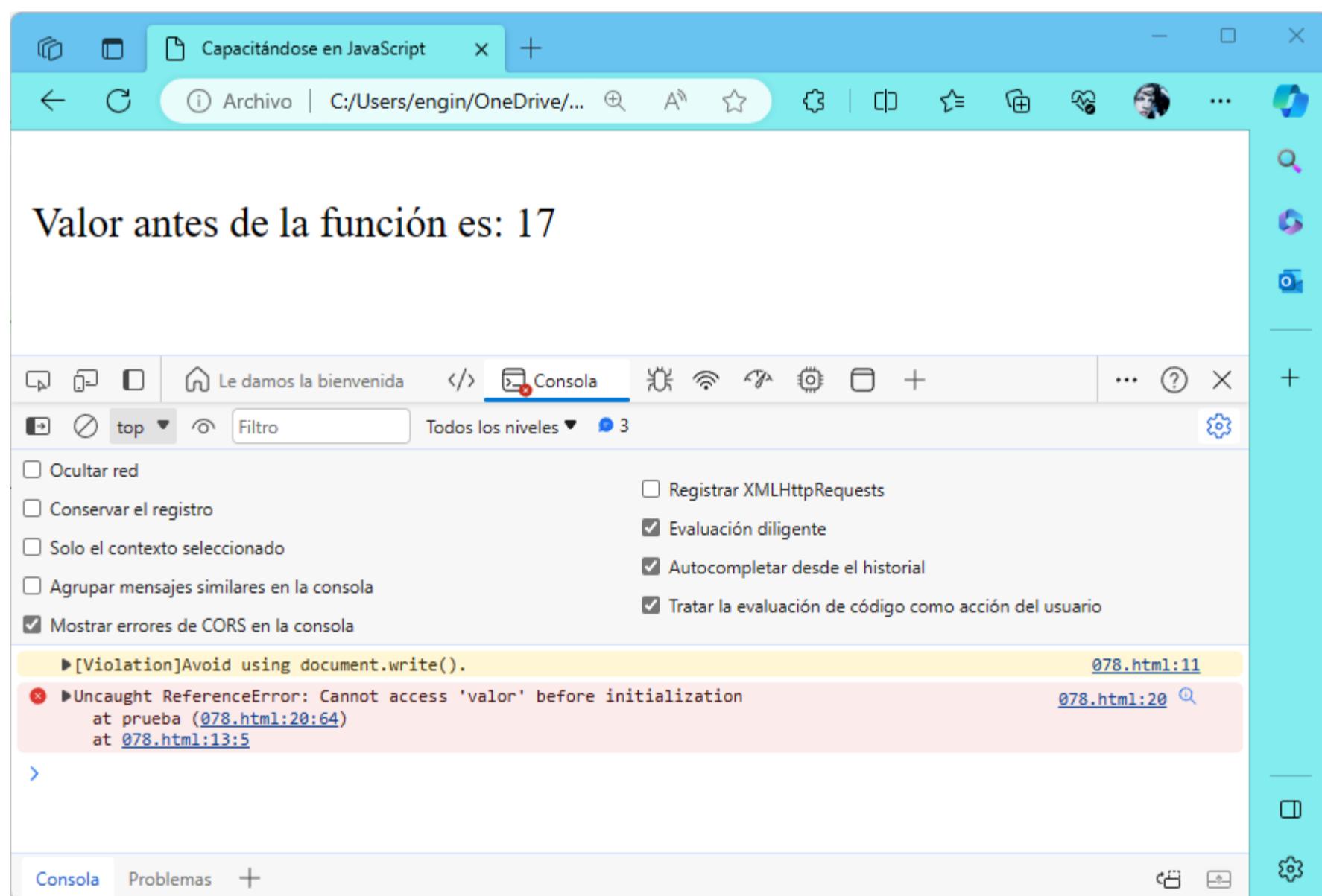


Ilustración 65: Ámbito (scope) de una variable

```

<!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

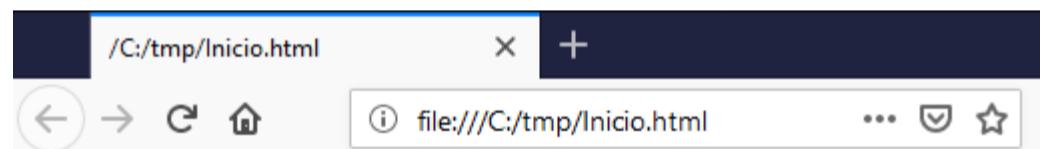
    //Imprime 17
    document.write("<br>Valor después de la función es: " + valor);

    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>

```



Valor antes de la función es: 17
 Valor dentro de la función es: 17
 Valor variable interna: 8906
 Valor después de la función es: 17

Ilustración 66: 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>
```

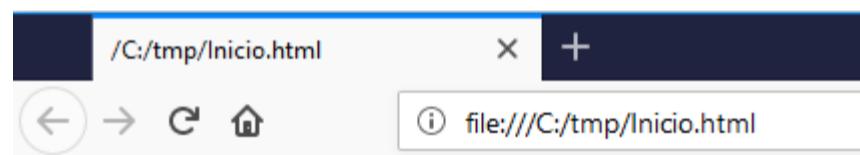


Ilustración 67: Cálculo de la factorial con función recursiva

```
<!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>
```



Ilustración 68: Cálculo del máximo común divisor con función recursiva

```
<!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>
```

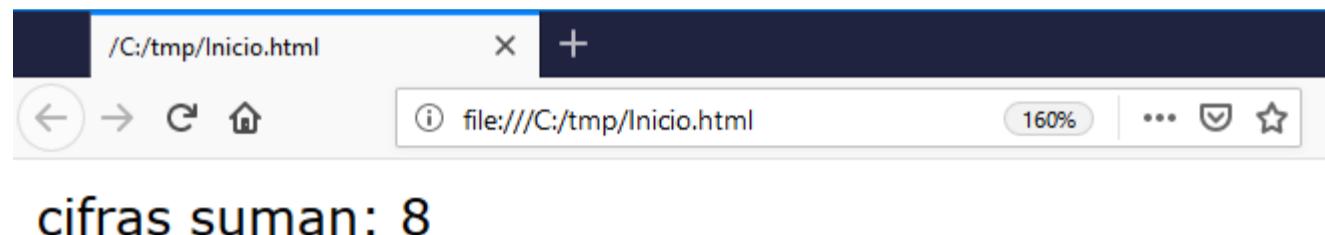
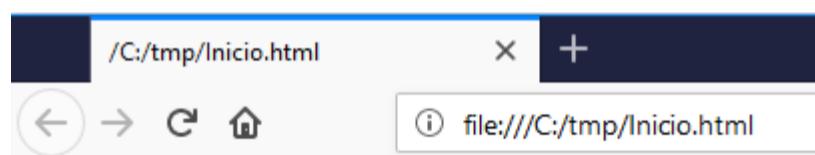


Ilustración 69: Suma de cifras con función recursiva

```
<!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>
```

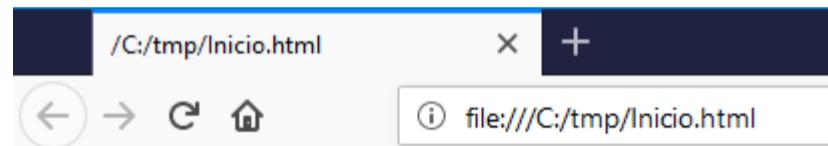


1000110

Ilustración 70: Conversión a número binario con función recursiva

```
<!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>
```



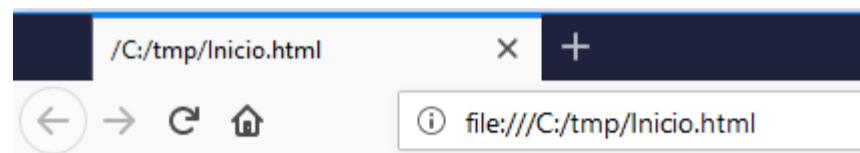
128

Ilustración 71: Cálculo de potencia con función recursiva

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>
```

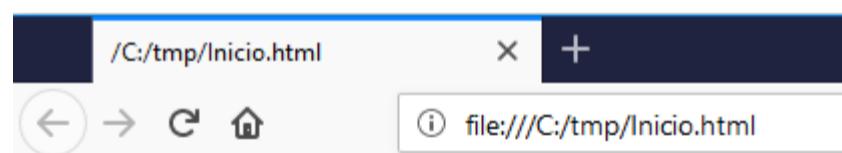


13

Ilustración 72: Eval como evaluador de expresiones

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>
```



1.198017100381385

Ilustración 73: Eval como evaluador de expresiones

```
<!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>
```

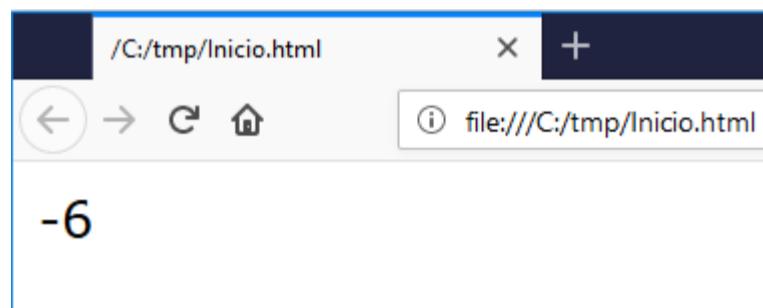


Ilustración 74: Eval como evaluador de expresiones

```
<!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>
```

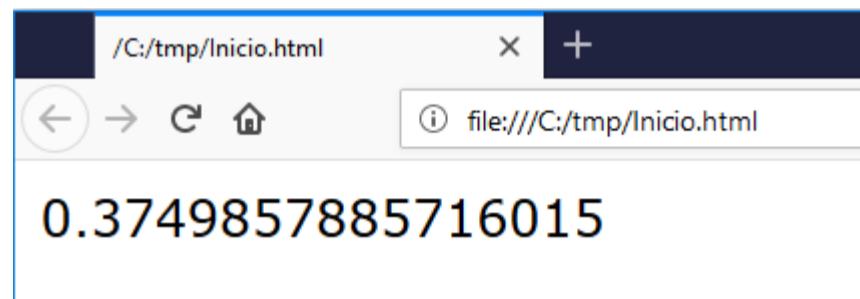


Ilustración 75: Número aleatorio

```
<!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>
```

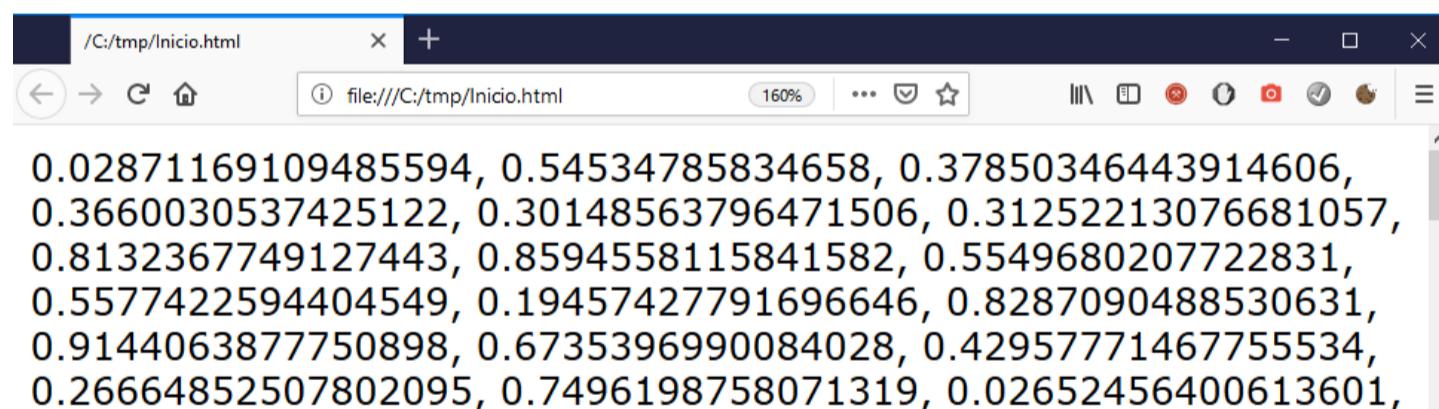


Ilustración 76: Listado de números aleatorios

```
<!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>
```

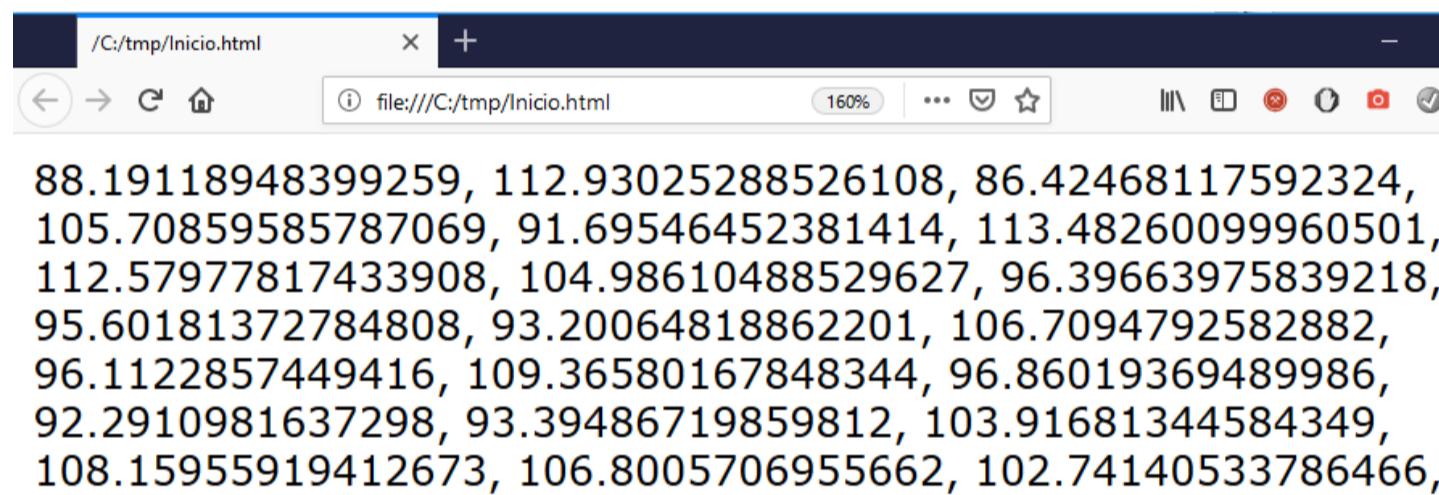


Ilustración 77: Distribución normal

```
<!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>
```

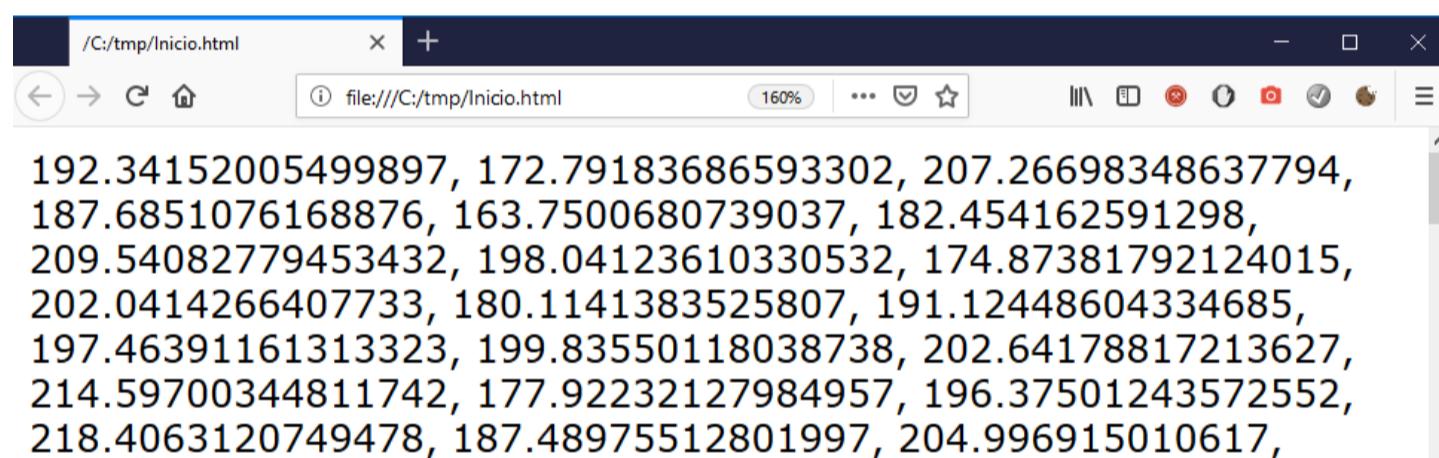
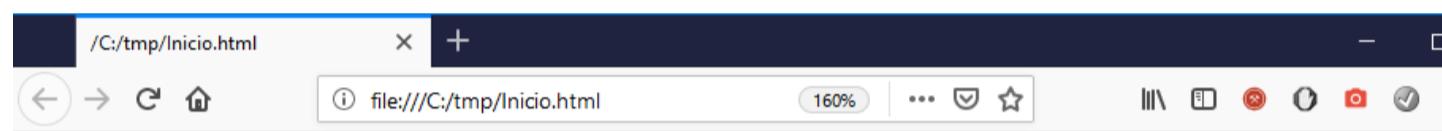


Ilustración 78: Distribución triangular

```
<!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>
```



24.074055060910958, 15.738809295288823, 32.73222112242828,
48.55045073392782, 18.07697909304522, 39.21438985067125,
43.291726207998934, 36.01663705766679, 43.5136552872398,
28.076865680530496, 12.622458628875378, 15.09721826831894,
35.44070876107921, 24.358365359386802, 21.8887709894043,
30.837152341377813, 20.462800855575072, 18.110379049814096,
12.787176231036664, 38.847595052491016, 46.746605385804386,
20.024111928546553, 15.266002603798663, 39.85230876591041,

Ilustración 79: Distribución uniforme

Manejo de errores con números

isNaN

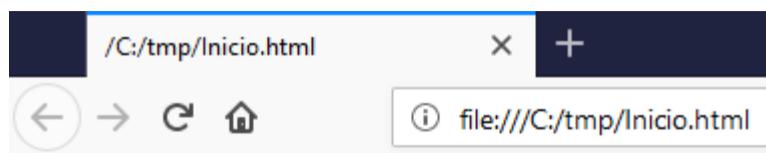
isNaN significa "is Not a Number", si lo que entra por parámetro no es un número retorna true. Más información: <https://es.wikipedia.org/wiki/NaN>

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>
```



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

Ilustración 80: Uso de isNaN

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>
```

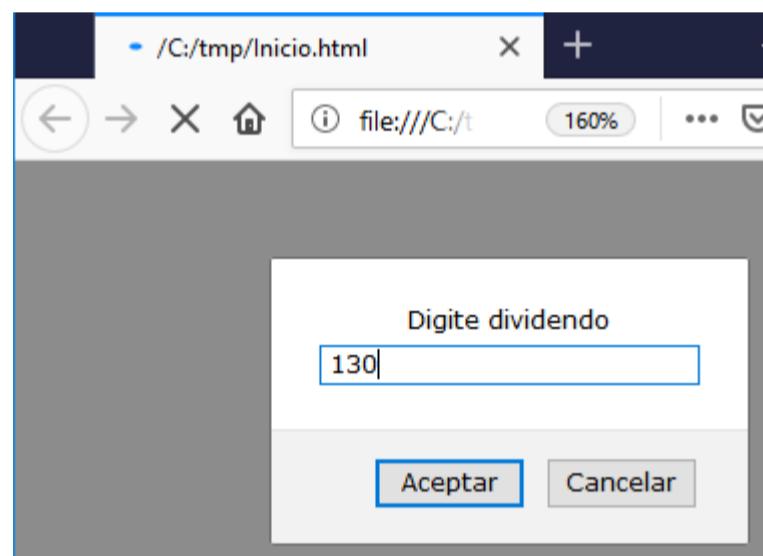


Ilustración 81: Ingresa dividendo

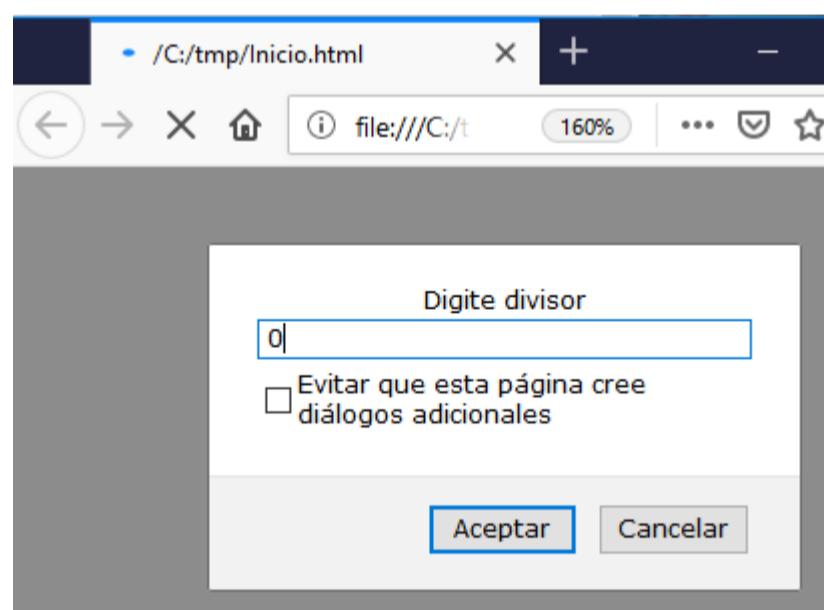


Ilustración 82: Ingresa divisor como cero para generar un error

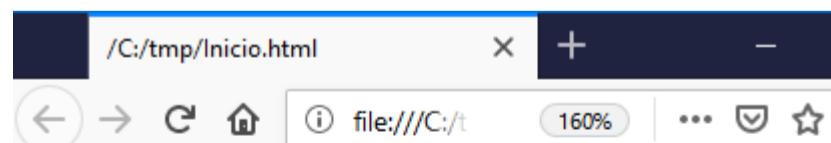


Ilustración 83: La división entre cero da "Infinity"

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>
```

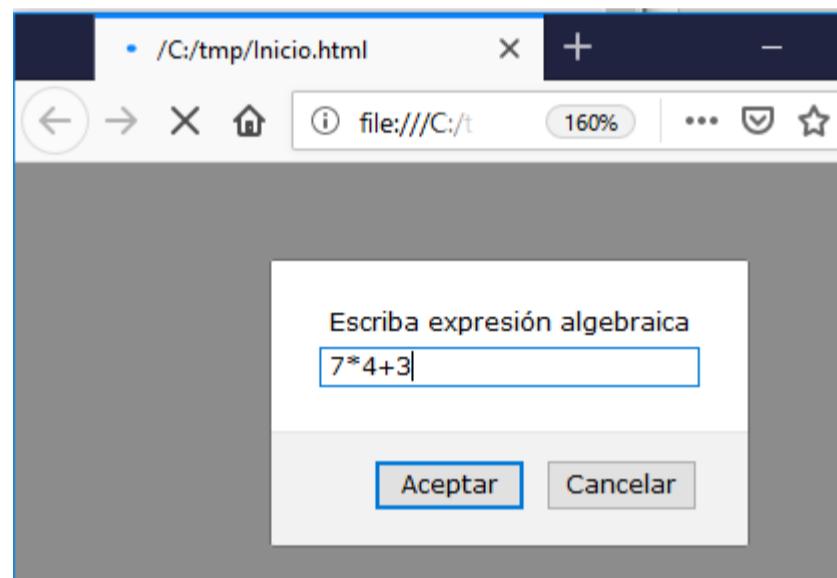


Ilustración 84: Ingresa expresión válida

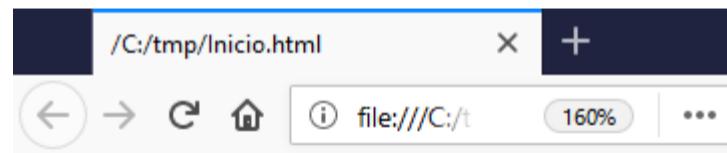


Ilustración 85: Hace el cálculo con expresión válida

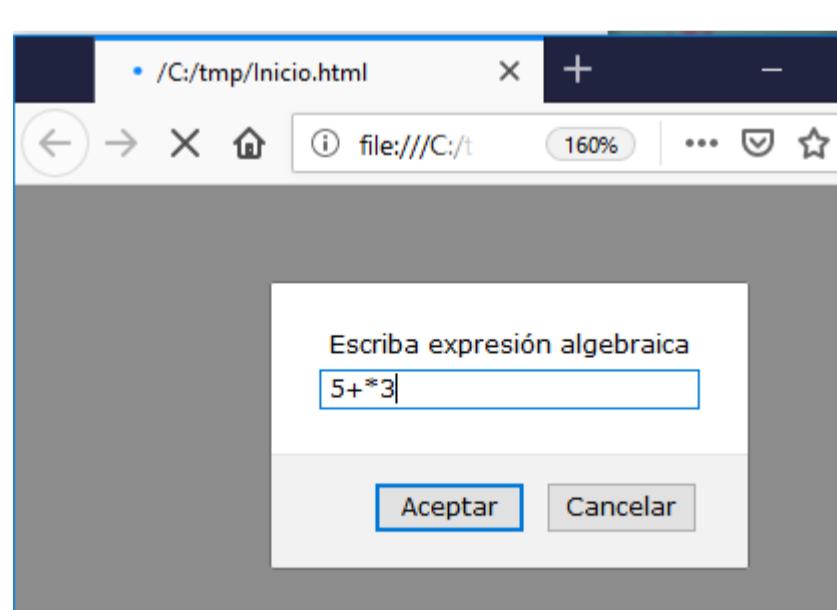
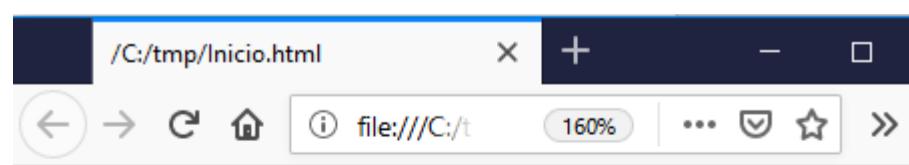


Ilustración 86: Ingresa expresión con sintaxis errónea



Mensaje de error: expected
expression, got '*'

Ilustración 87: 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>
```

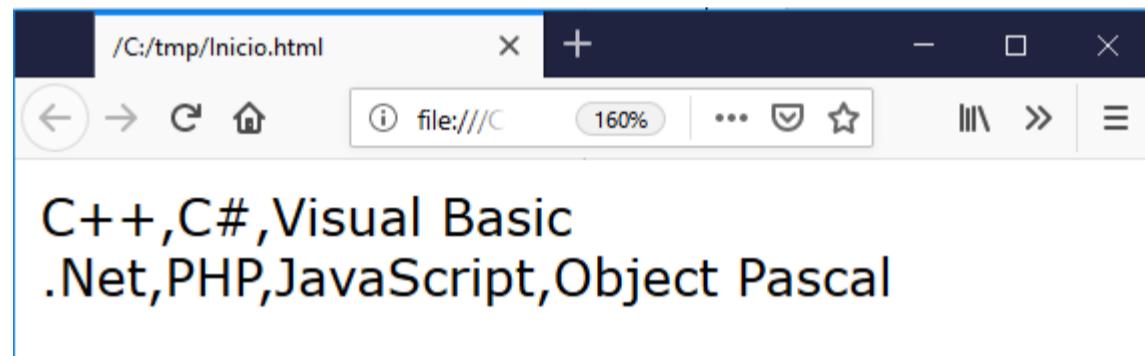


Ilustración 88: Arreglo unidimensional impreso

```
<!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>
```

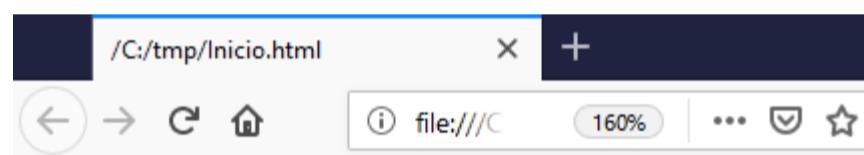


Ilustración 89: Muestra ítem a ítem el arreglo unidimensional

```
<!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>
```

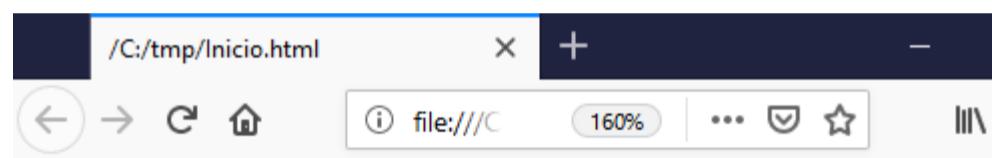


Ilustración 90: Otra forma de declarar arreglo unidimensional

```
<!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>
```

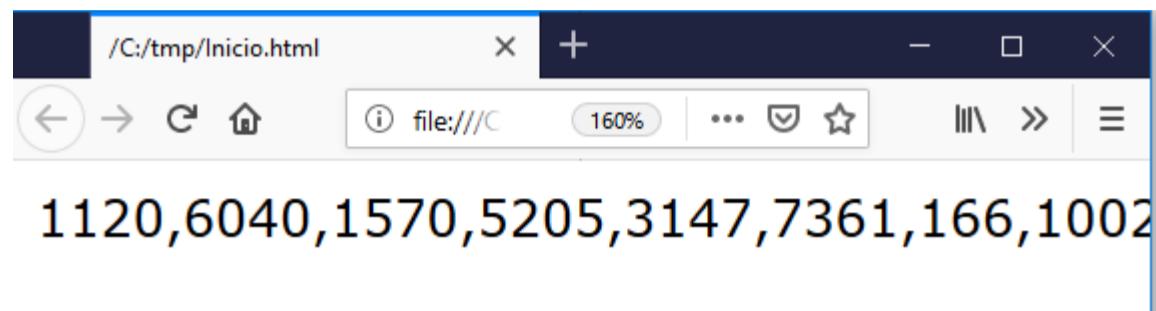


Ilustración 91: Otra forma de declarar arreglo unidimensional

```
<!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>
```



Ilustración 92: Uso de splice() para borrar ítems en un arreglo

```
<!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>
```

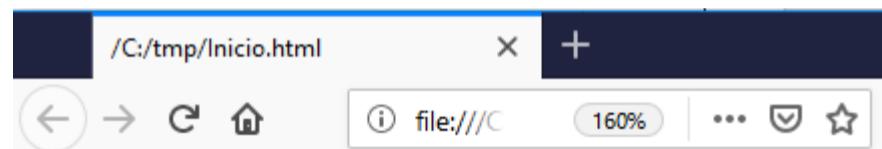


Ilustración 93: Operación matemática con elementos del arreglo unidimensional

```
<!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>
```

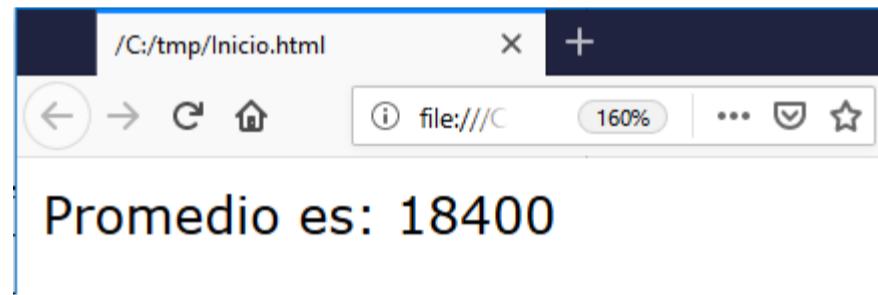


Ilustración 94: Operación matemática con elementos del arreglo unidimensional

```
<!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>
```

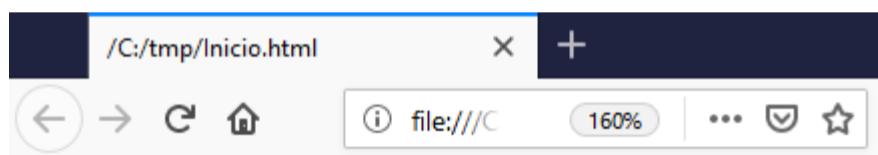


Ilustración 95: Operación matemática con elementos del arreglo unidimensional

```
<!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>
```

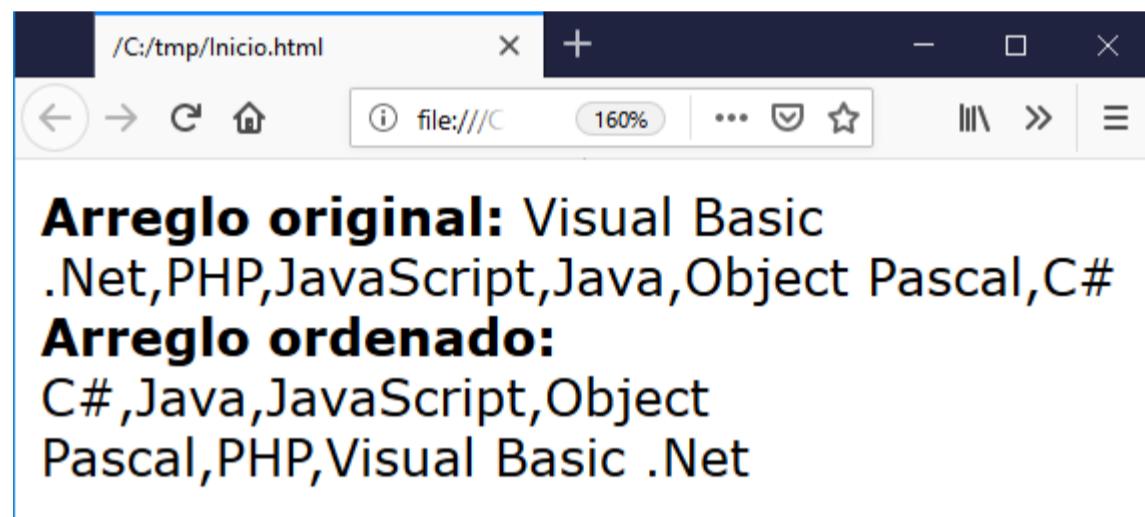


Ilustración 96: Ordenación de arreglo unidimensional

```
<!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>
```

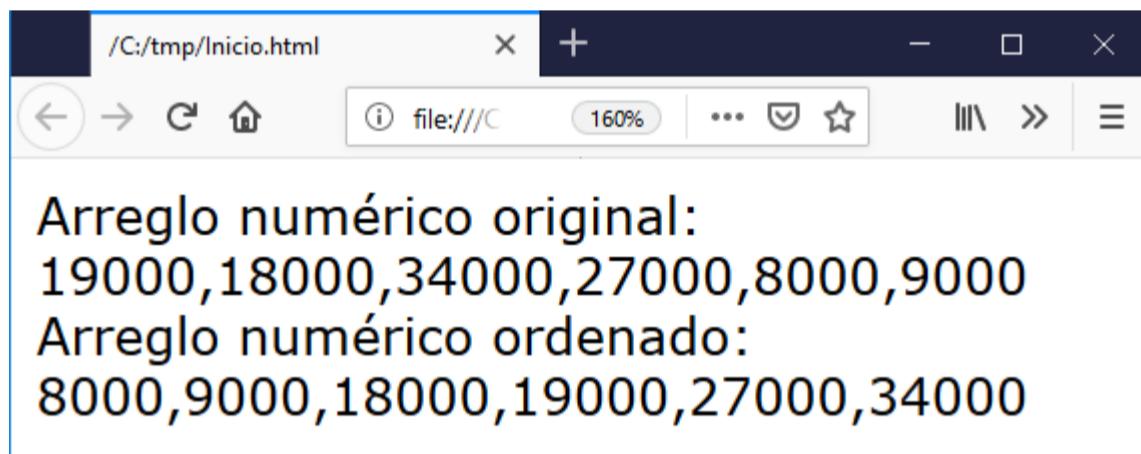


Ilustración 97: Ordenación de arreglo unidimensional

```

<!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>

```

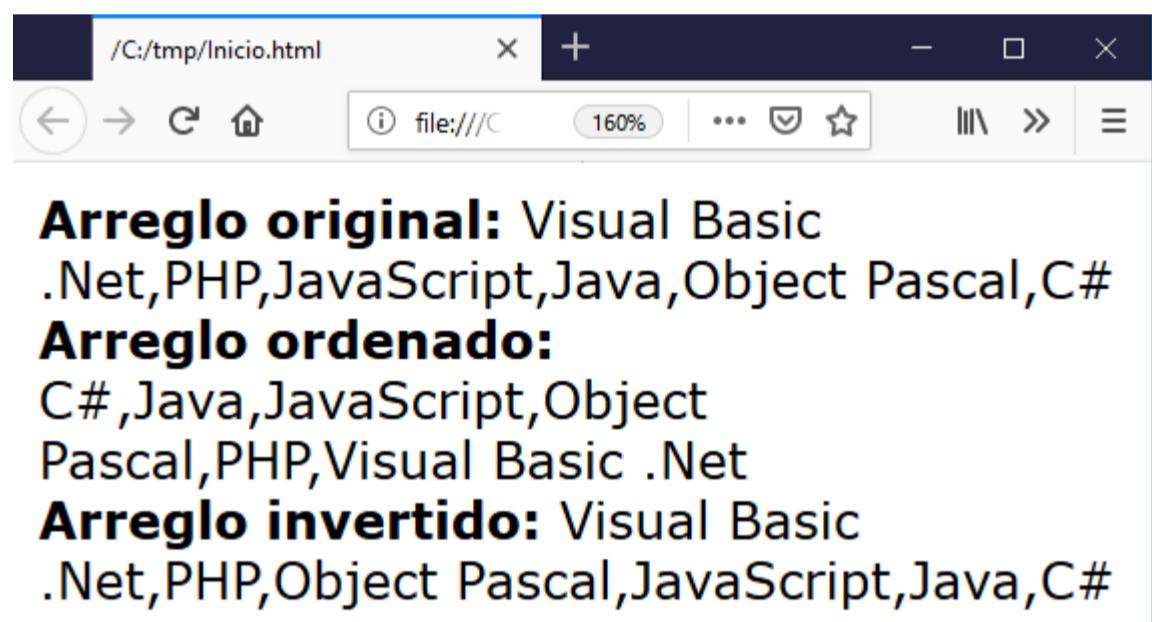


Ilustración 98: Ordenación de arreglo de menor a mayor, y de mayor a menor

```
<!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>
```

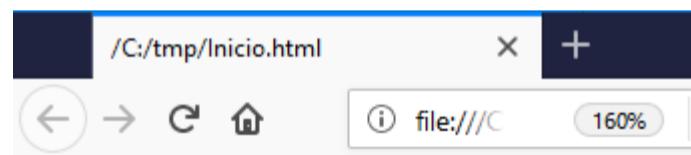


Ilustración 99: Dado un arreglo extraer los valores únicos

```

<!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>

```

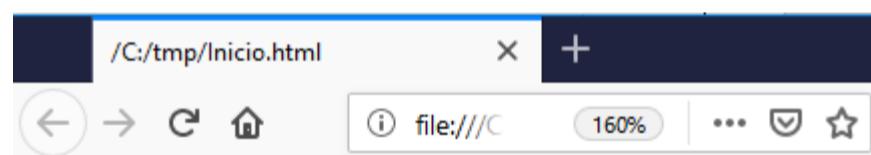
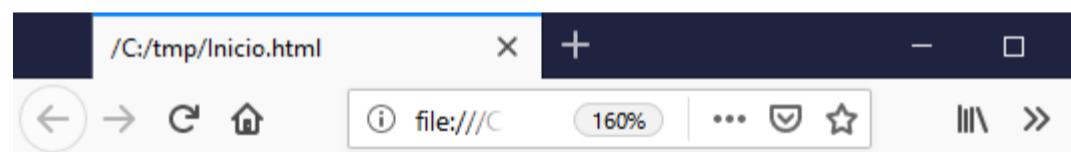


Ilustración 100: ¿Función genérica para hallar menor y mayor valor?

```
<!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>
```



Pares: 10,20,24,2,8,6,16,2,6
Impares: 13,23,23,29,5,19,27

Ilustración 101: Separe los elementos pares e impares de una lista, y retorne ambas listas

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>
```

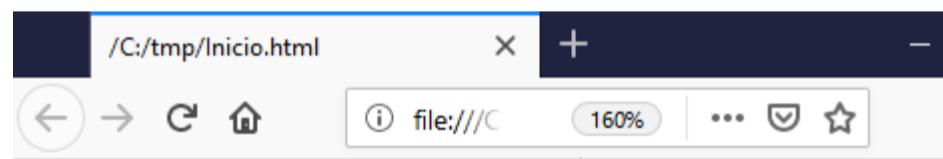
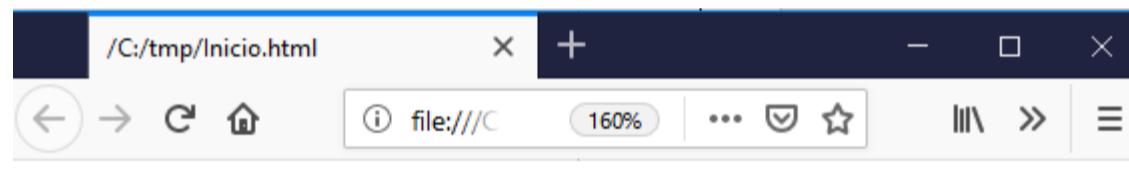


Ilustración 102: Ordenación por el método de burbuja

```
<!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>
```



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

Ilustración 103: Ordenación por burbuja

```

<!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>

```

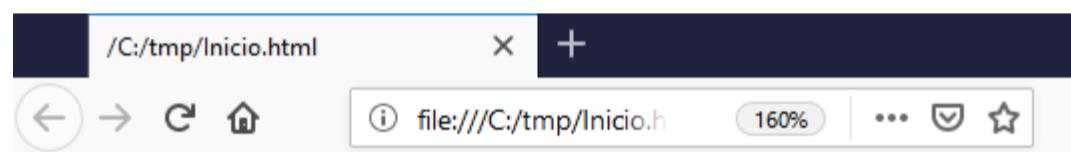


Ilustración 104: Ordenación por método de burbuja mejorado

```
<!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>
```

```
<!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>
```

```
<!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>
```

```
<!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>
```

```
<!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 = [], pivot = arreglo[0];
    for (let i = 1; i < arreglo.length; i++) {
        if (arreglo[i] < pivot)
            izquierdo.push(arreglo[i]);
        else
            derecho.push(arreglo[i]);
    }
    return quicksort(izquierdo).concat(pivot, quicksort(derecho));
}
</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, 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>
```

```
<!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>
```

```
<!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>
```

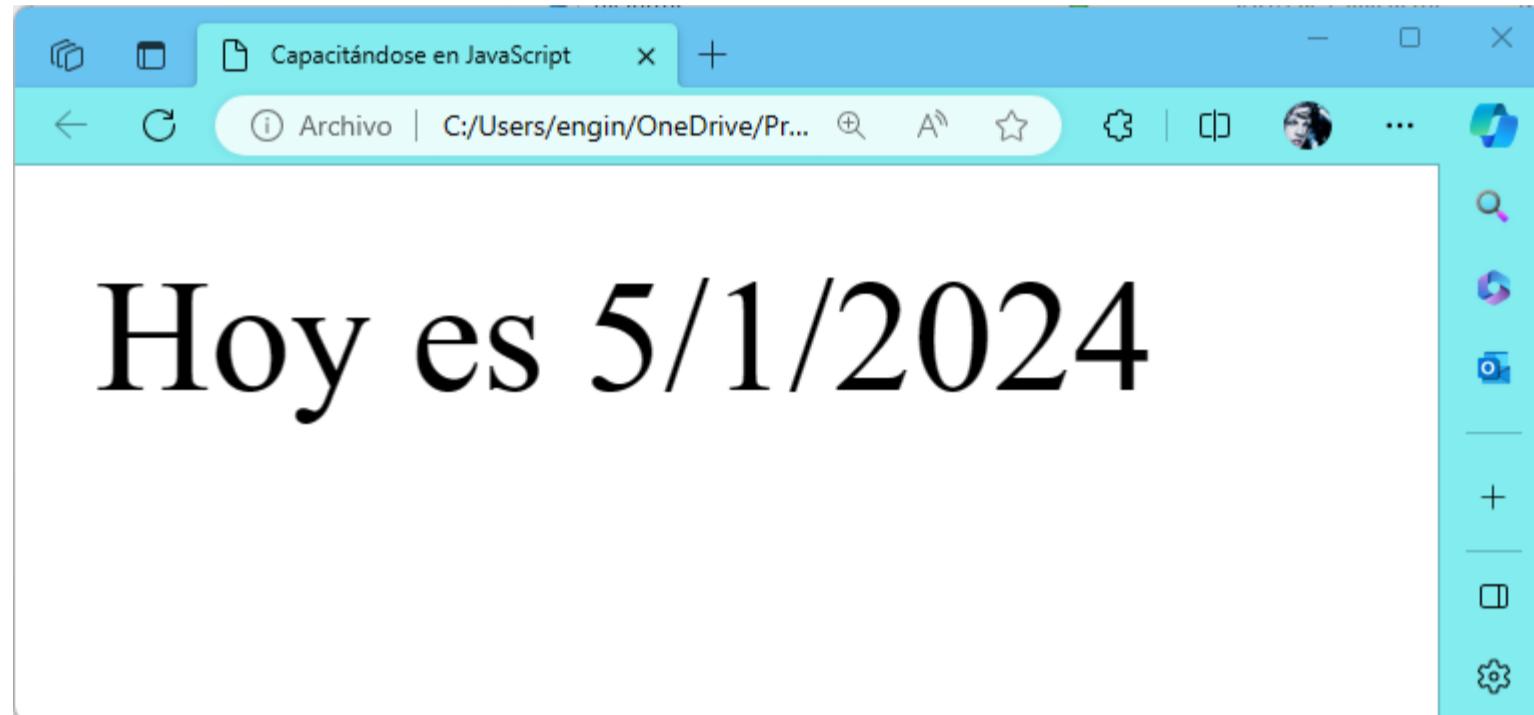


Ilustración 105: Fecha actual

```
<!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>
```



Ilustración 106: Fecha actual

```
<!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 diaSemana = fecha.getDay();
    let mesNombre = meses[mes]; //Convierte el valor numérico en nombre de mes
    let diaNombre = diasemana[diaSemana]; //Convierte el valor numérico en nombre de día
    document.write(diaNombre + ", " + dia + " de " + mesNombre + " de " + año);
</script>
</body>
</html>
```

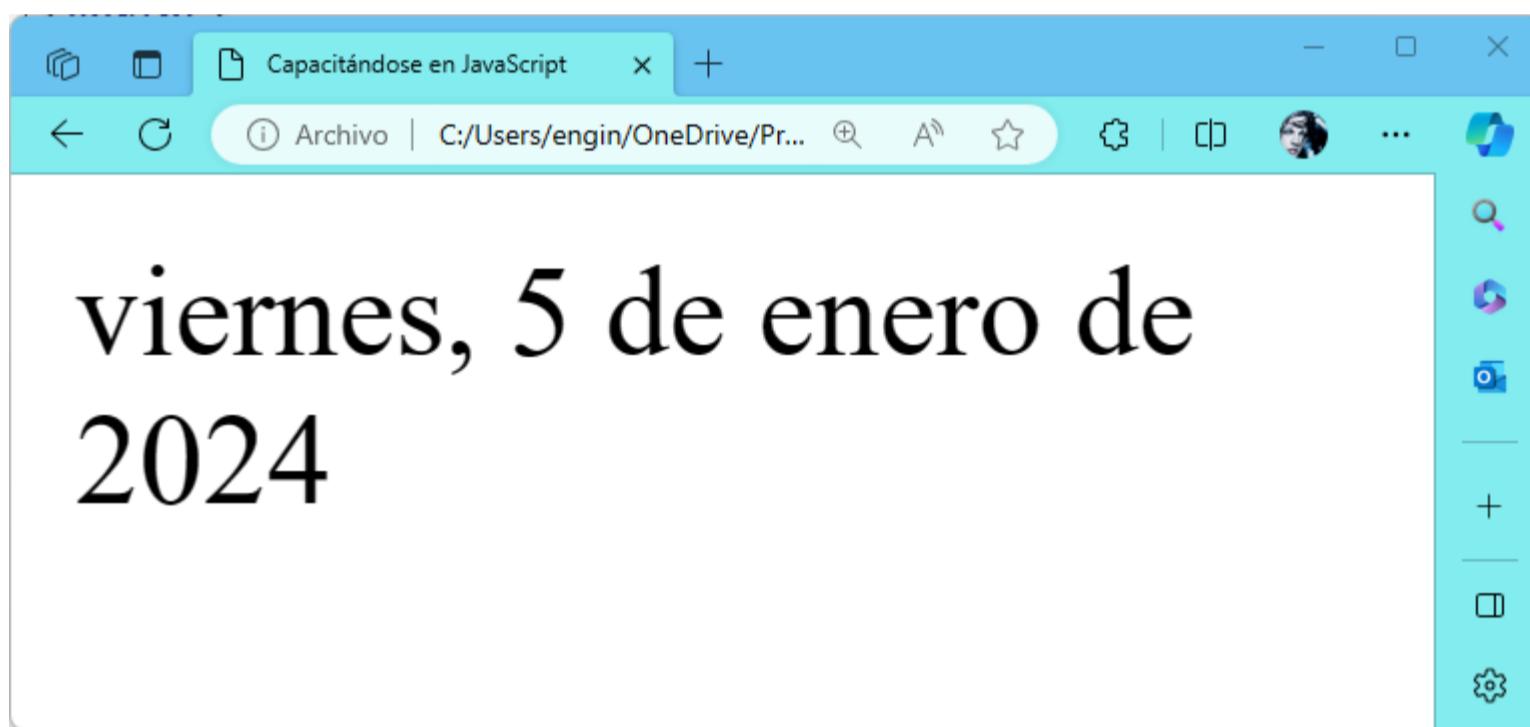
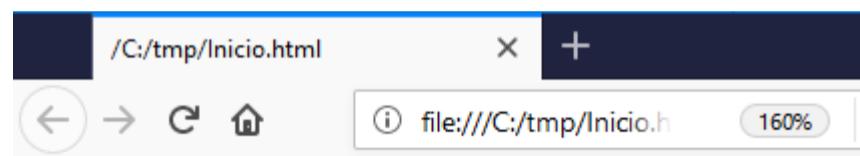


Ilustración 107: Fecha actual

```
<!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>
```



Dragón de Komodo

Ilustración 108: Concatenación de cadenas

```
<!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>
```

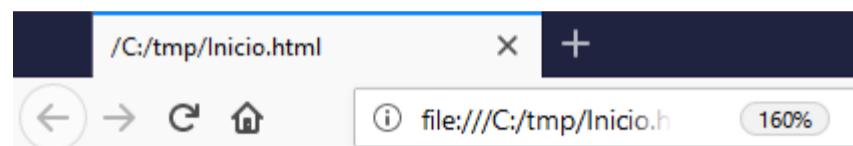


Ilustración 109: Concatenación de números y cadenas

```
<!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>
```

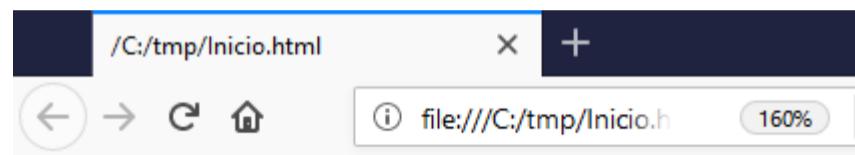


Ilustración 110: Primera letra de una cadena

```
<!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>
```

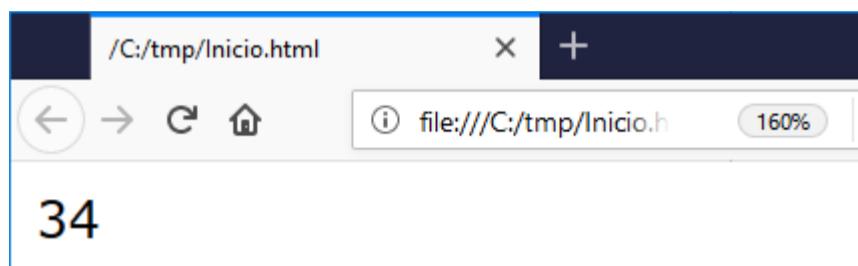


Ilustración 111: Longitud de una cadena

```
<!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>
```

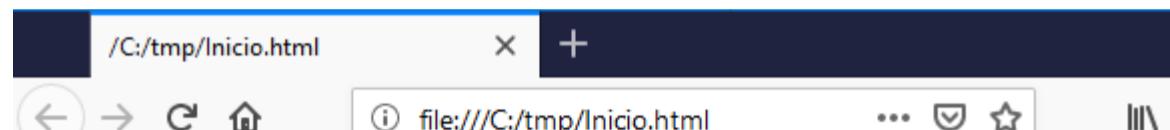


Ilustración 112: Convertir a mayúsculas y minúsculas

```
<!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>
```

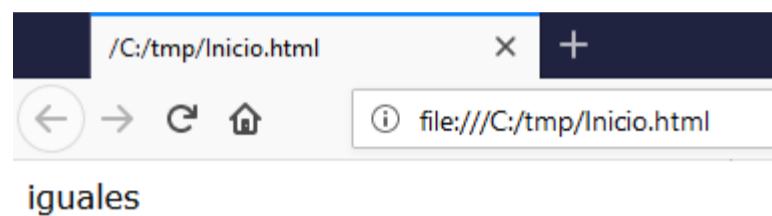


Ilustración 113: Comparación de cadenas

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>
```

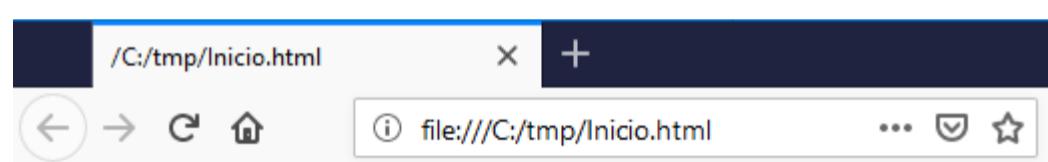


Ilustración 114: Invierte la cadena

Quitar los espacios de una cadena

133.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>
```



Ilustración 115: Quitar los espacios de una cadena

Quitar las vocales de una cadena

134.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 = "aeiouAEIOUÁÉÍÓÚáéíóúäéöü";
    for (let letra = 0; letra < vocales.length; letra++)
        if (caracter === vocales[letra])
            return true;

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

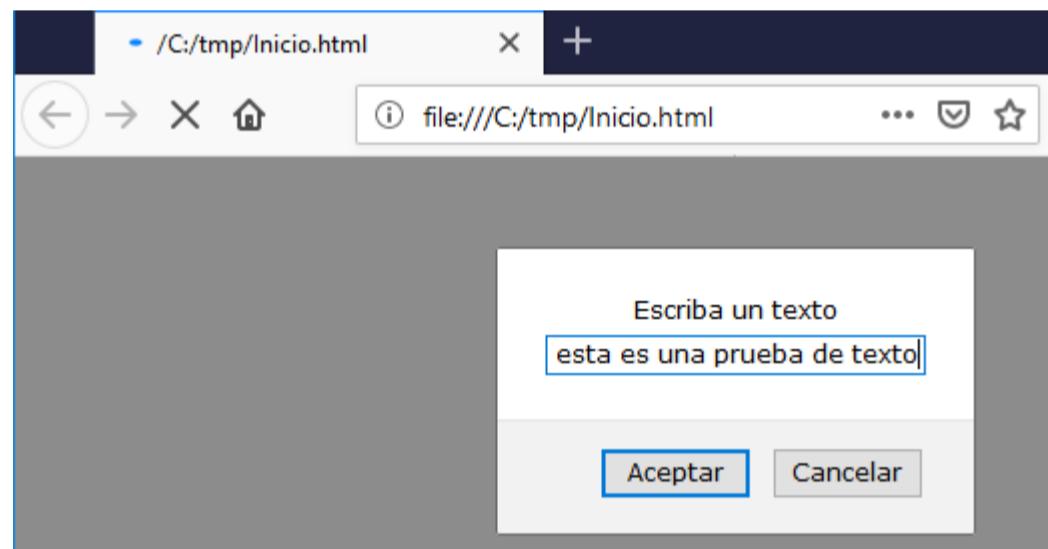


Ilustración 116: Quitar vocales (minúsculas, mayúsculas, tildadas)

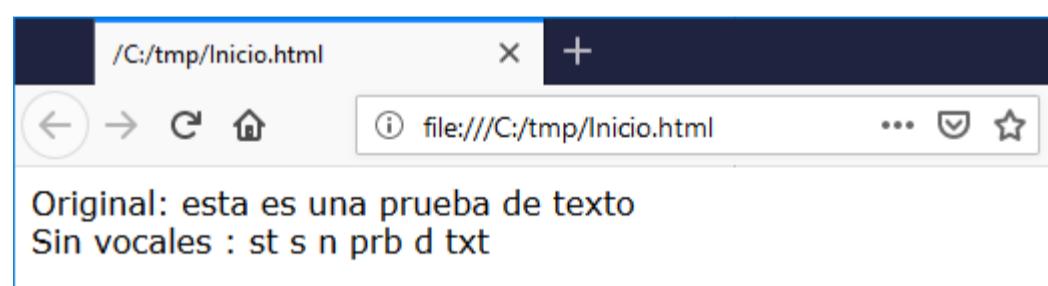


Ilustración 117: Quitar vocales (minúsculas, mayúsculas, tildadas)

```
<!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 += "abcdefghijklmnopqrstuvwxyz";
        permite += "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        permite += " 1234567890";
        for (let letra = 0; letra < permite.length; letra++)
            if (caracter === permite[letra])
                return true;
        return false;
    }
</script>
</body>
</html>
```

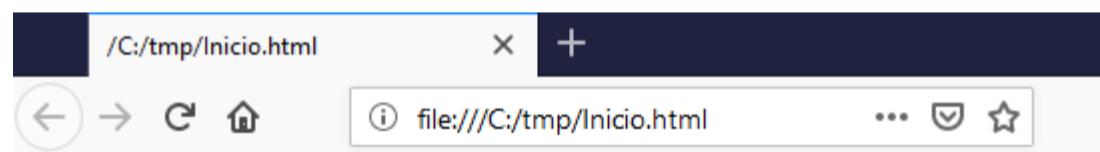


Ilustración 118: Retirar caracteres no permitidos de una cadena

```

<!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>

```

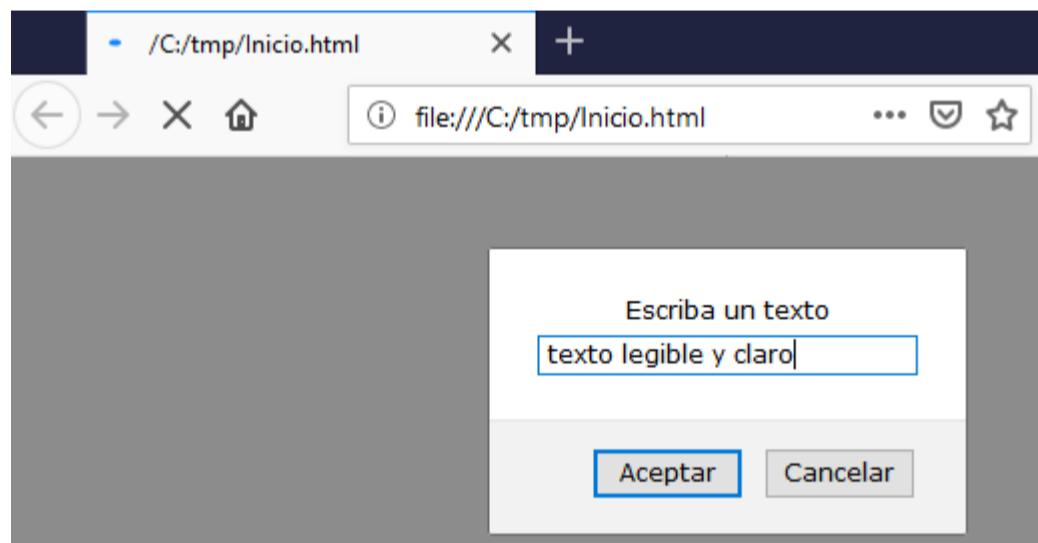


Ilustración 119: Cifrado sencillo

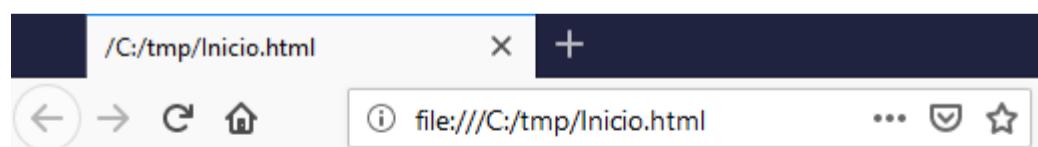


Ilustración 120: Cifrado sencillo

```

<!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>

```

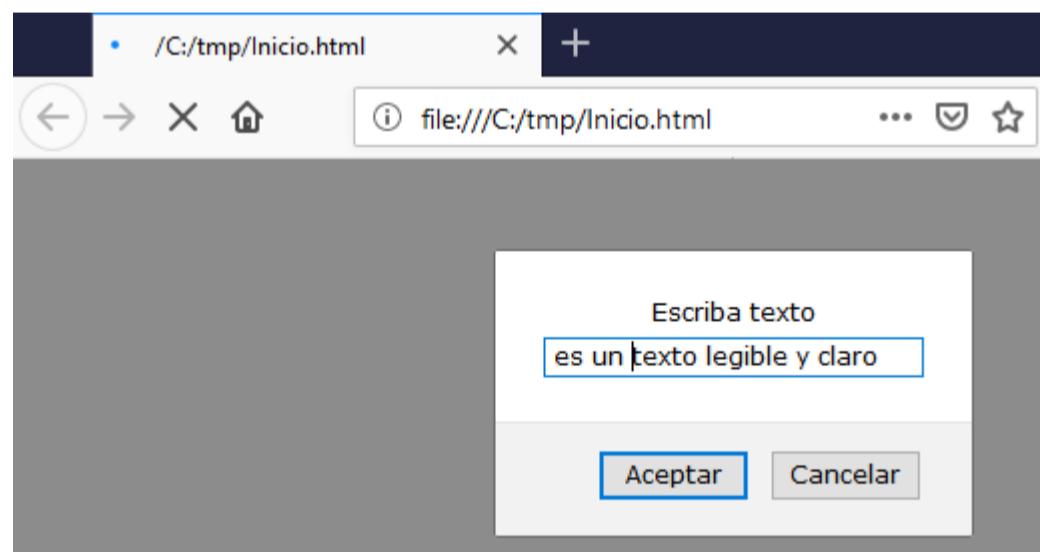


Ilustración 121: Cifrado sencillo con clave de cifrado

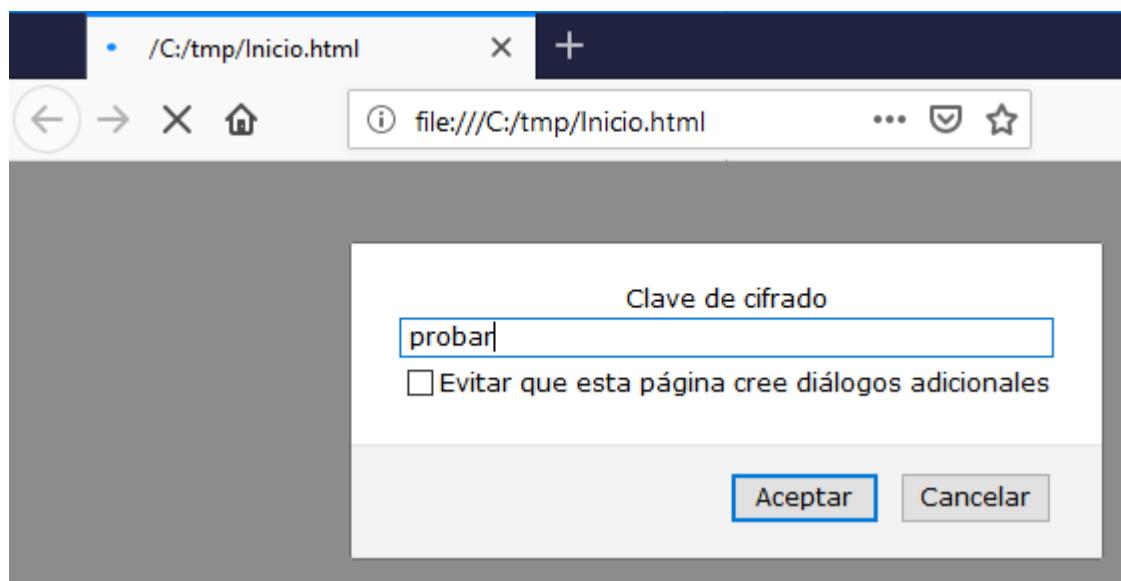


Ilustración 122: Cifrado sencillo con clave de cifrado

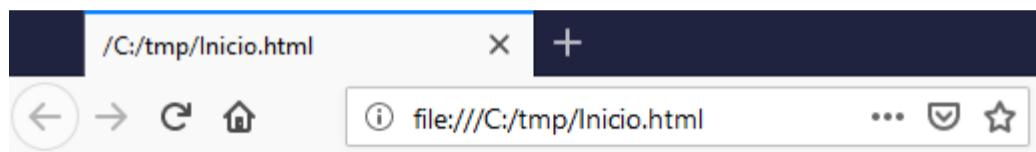


Ilustración 123: Cifrado sencillo con clave de cifrado

```

<!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 = "888888888888888888"; //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 + " + " + numeroB + "-----<br>" + resultado);
    }
</script>
</body>
</html>

```

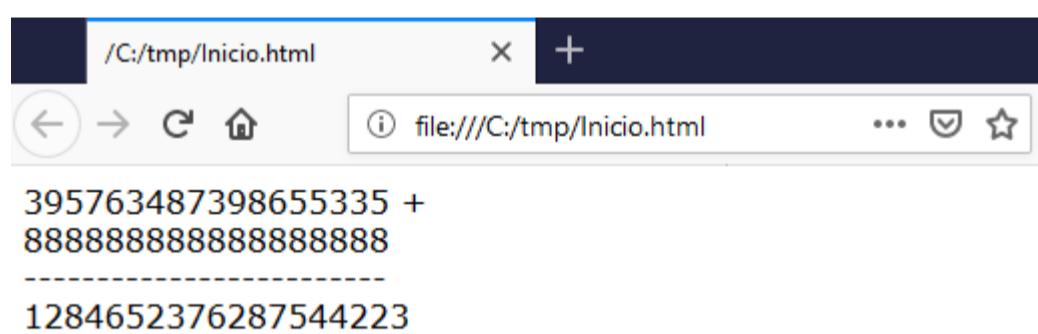


Ilustración 124: Suma dos números enteros positivos almacenados en cadenas

```
<!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>
```

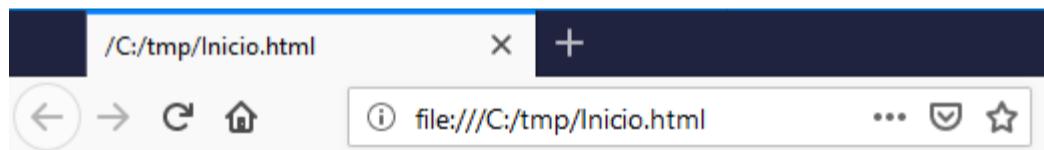


Ilustración 125: Crea el arreglo bidimensional

```

<!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>

```

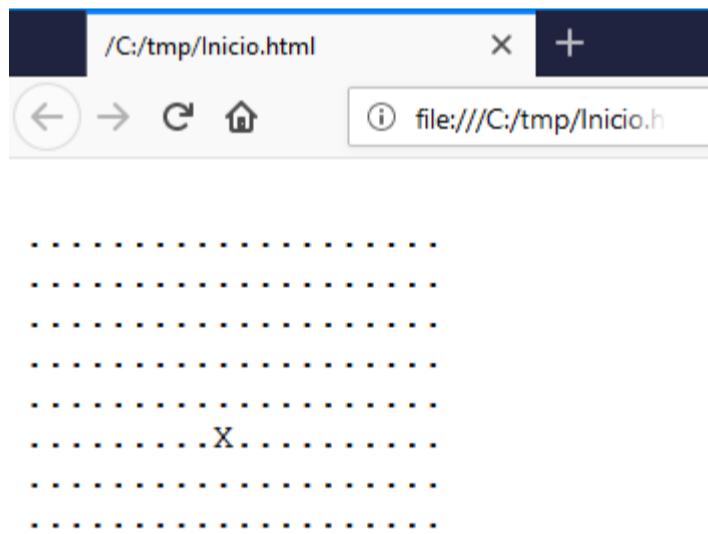


Ilustración 126: Pone una X en una posición al azar

```

<!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>

```

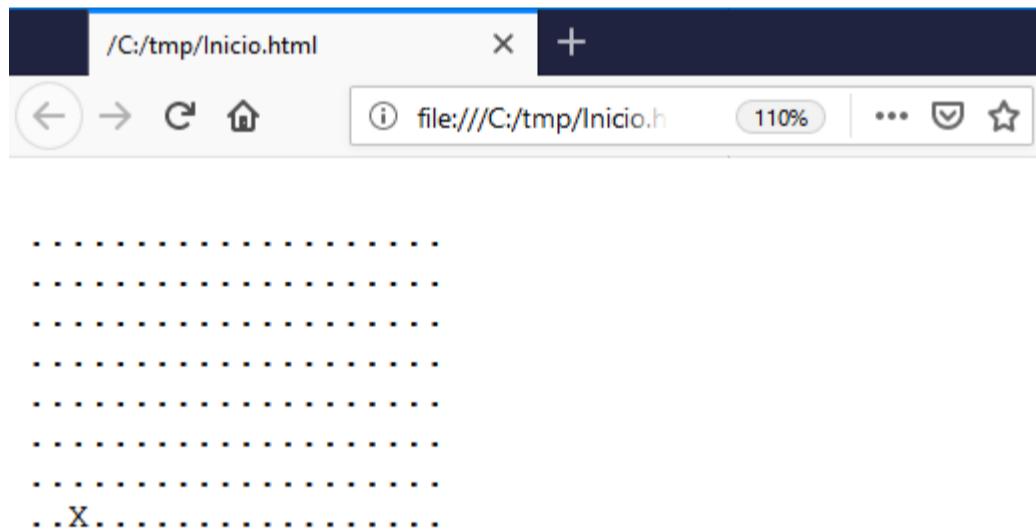


Ilustración 127: Función genérica para crear arreglo bidimensional

```

<!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>

```

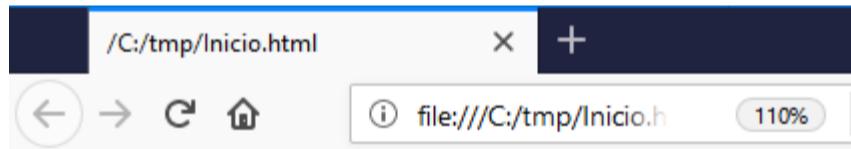


Ilustración 128: Función genérica para crear arreglo bidimensional

Ejemplos de programas usando arreglos bidimensionales

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

143.html

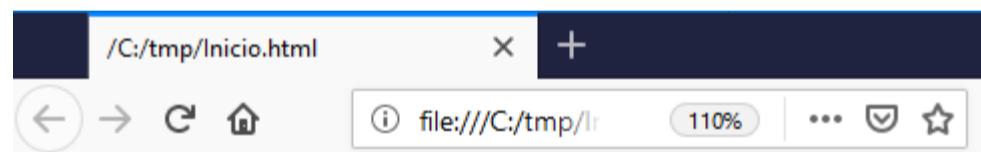
```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Crear un tablero de ajedrez de 8*8, poner en una posición al
       azar una Torre y mostrar su ataque */

    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>
```



```
...x.....
...x.....
...x.....
...x.....
...x.....
xxTxxxxx
...x.....
...x.....
```

Ilustración 129: Poner en una posición al azar la torre y mostrar su ataque

```

<!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>

```

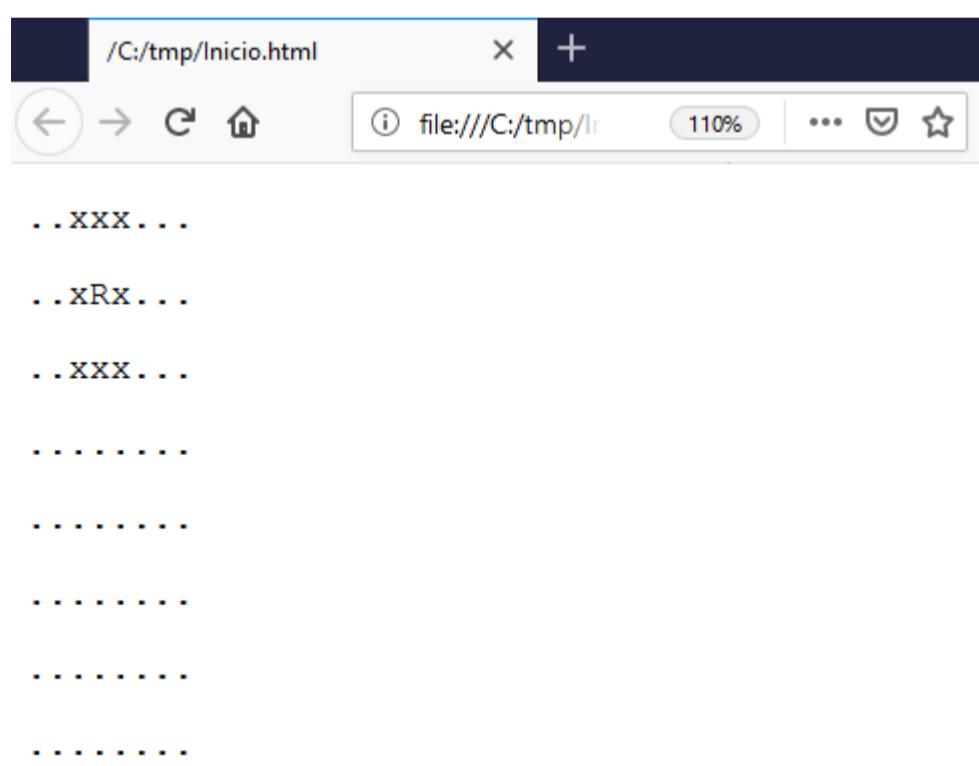


Ilustración 130: Poner un Rey al azar en un tablero de ajedrez y mostrar su ataque

```

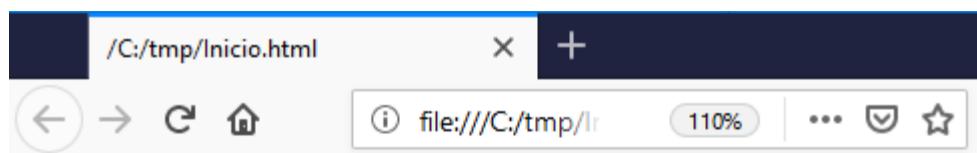
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Crear un tablero de ajedrez de 8*8, poner en una posición al
       azar un Alfil y mostrar su ataque */
    let tabla = GeneraArregloBidimensional(8, 8, '.');
    PoneAlfilAzar(tabla);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

```



.....

.....X

X.....X.

.X...X..

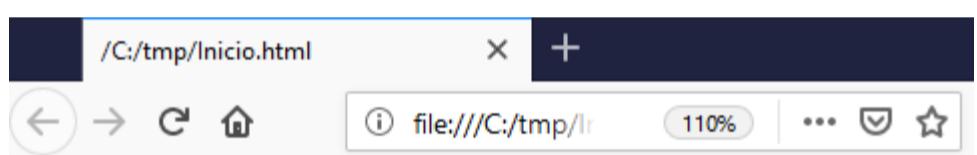
..X.X...

...A....

..X.X...

.X...X..

Ilustración 131: Poner un alfil al azar en un tablero de ajedrez y mostrar su ataque



...X.X..

...A....

..X.X..

.X...X..

.X.....X

X.....

.....

.....

Ilustración 132: Poner un alfil al azar en un tablero de ajedrez y mostrar su ataque

```

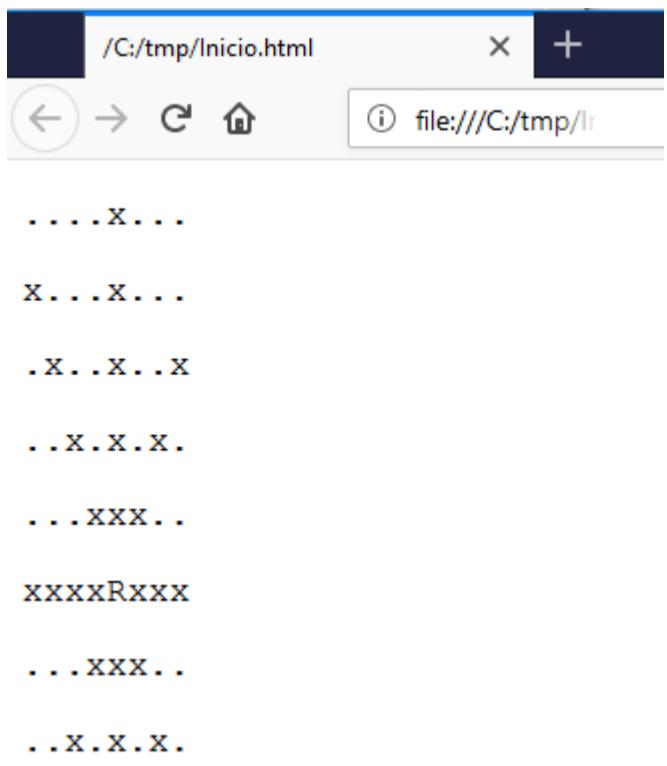
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    /* Crear un tablero de ajedrez de 8*8, poner en una posición al
       azar la Reina y mostrar su ataque */
    let tabla = GeneraArregloBidimensional(8, 8, '.');
    PoneReinaAzar(tabla);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

```

 /C:/tmp/Inicio.html X +
 ⏪ ⏹ ⏺ ⏻ ⏻ ⓘ file:///C:/tmp/|r

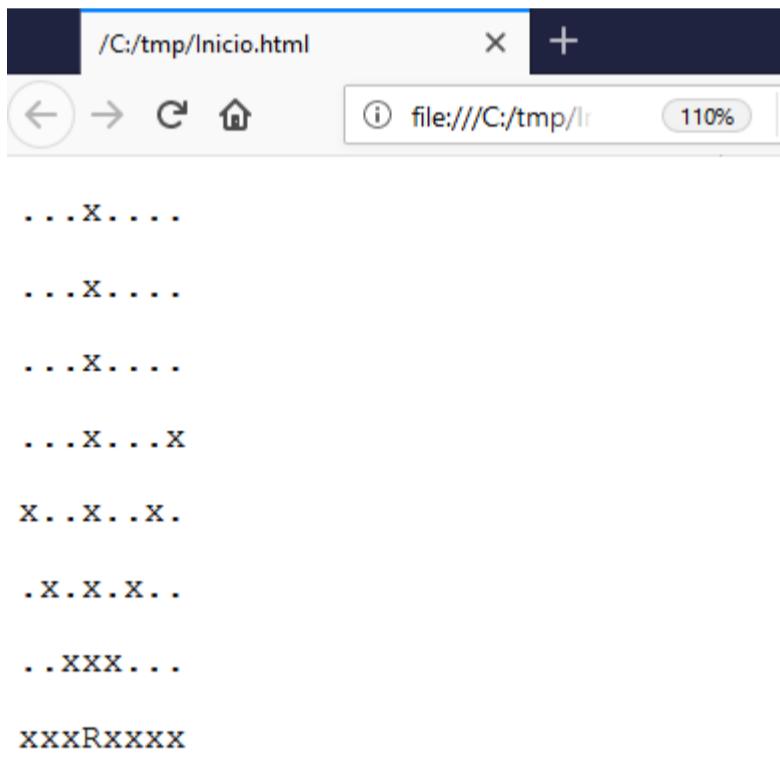
....X....
X...X...
.X..X..X
.X.X.X.
.X.XX..

xxxxRxXX

.X.XX..

.X.X.X.

Ilustración 133: Poner una Reina al azar en un tablero de ajedrez y mostrar su ataque

 /C:/tmp/Inicio.html X +
 ⏪ ⏹ ⏺ ⏻ ⏻ ⓘ file:///C:/tmp/|r 110%

....X....
....X....
....X....
....X....X
X..X..X.
.X..X..X.
.X.XX...

XXXRxXXXX

Ilustración 134: Poner una Reina al azar en un tablero de ajedrez y mostrar su ataque

Ruta del menor costo

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

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 minValor = 15; //Valor mínimo que tendrá ir de una ciudad a otra
  let maxValor = 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, minValor, maxValor);

  //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, minValor, maxValor) {
    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 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>

```

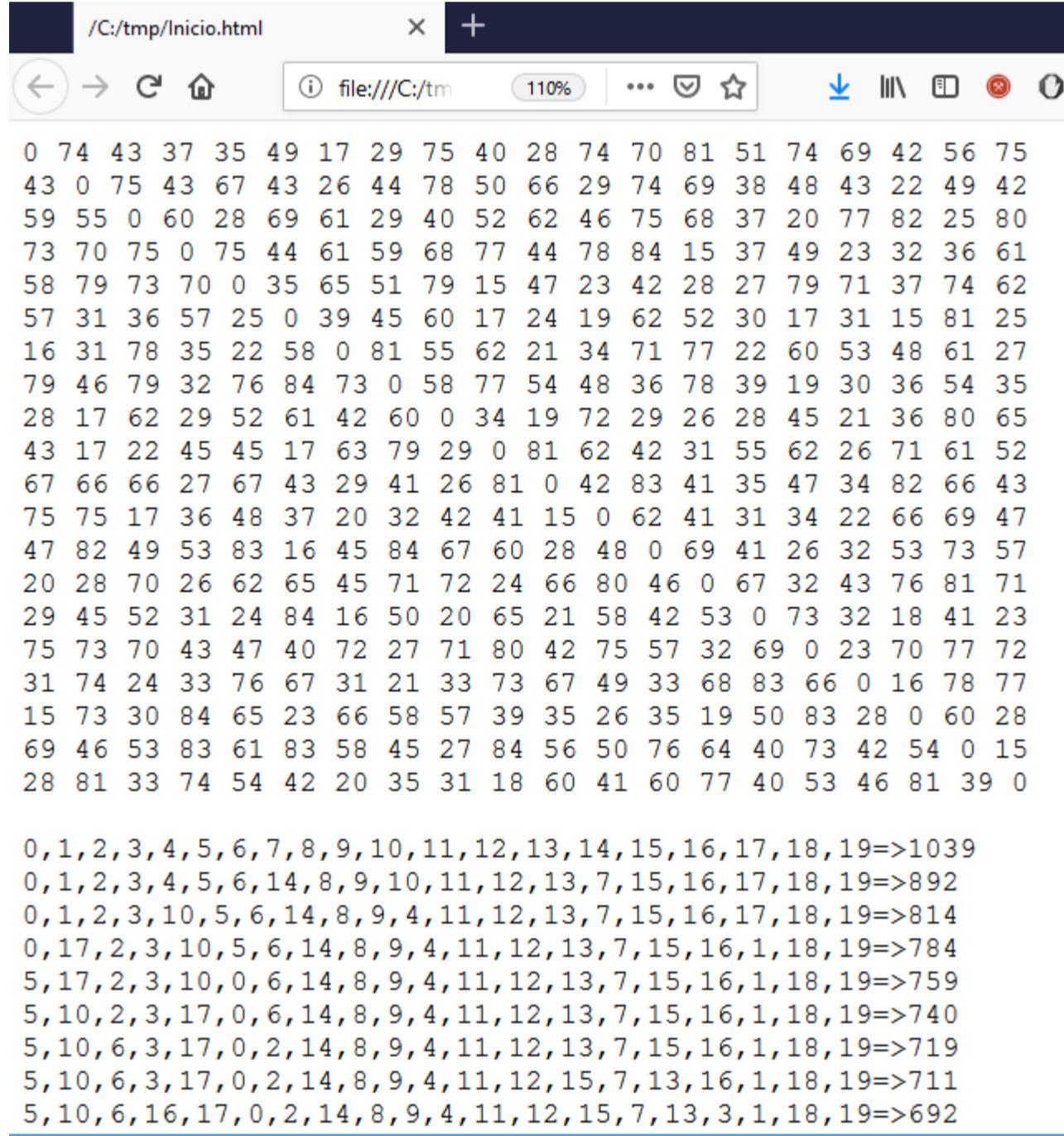


Ilustración 135: Ruta del menor costo. Genera nuevas rutas más económicas

```

<!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 */

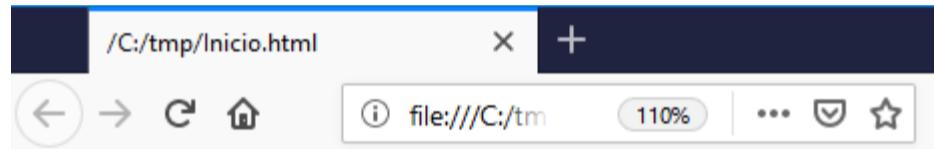
```

```

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>

```



Ciclos totales: 14762

5 3 4 6 7 8 9 1 2

6 7 2 1 9 5 3 4 8

1 9 8 3 4 2 5 6 7

8 5 9 7 6 1 4 2 3

4 2 6 8 5 3 7 9 1

7 1 3 9 2 4 8 5 6

9 6 1 5 3 7 2 8 4

2 8 7 4 1 9 6 3 5

3 4 5 2 8 6 1 7 9

Ilustración 136: 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>
```

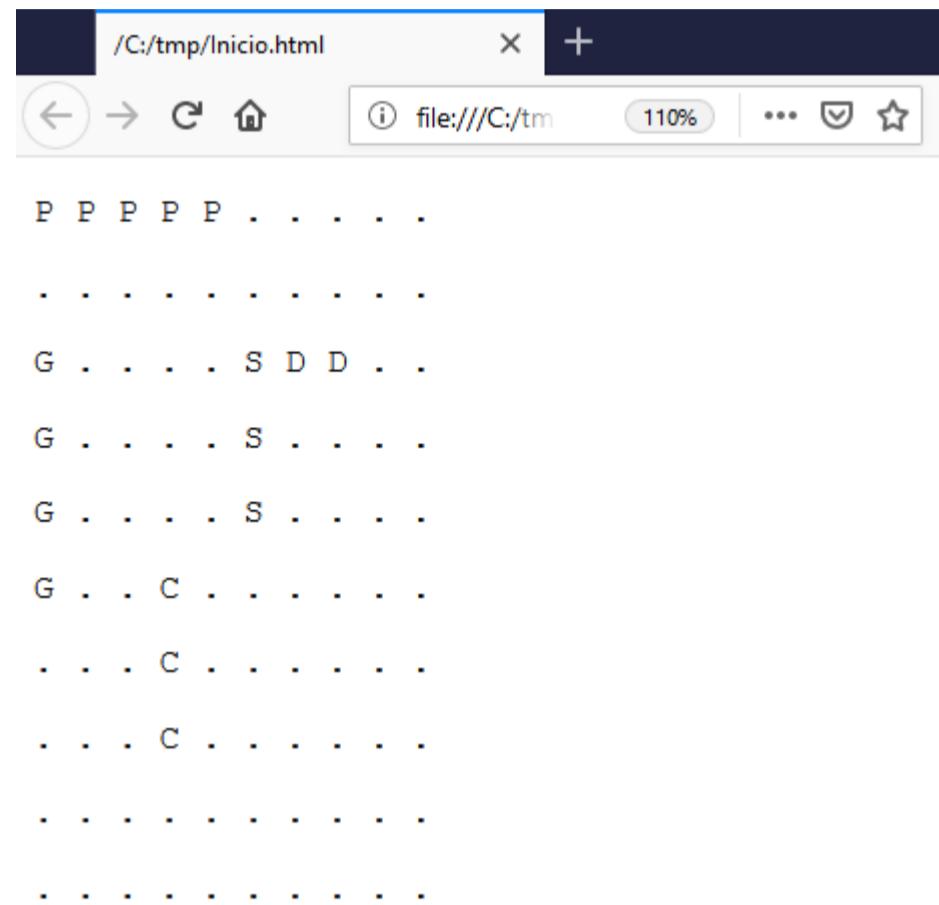
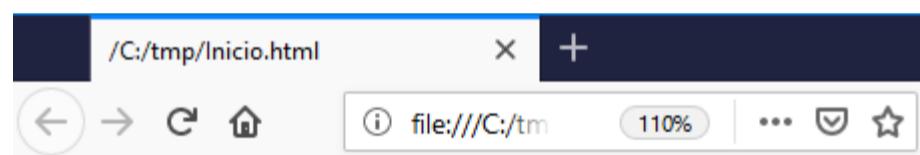


Ilustración 137: Barcos ubicados en el juego de mesa Batalla Naval

```
<!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>
```

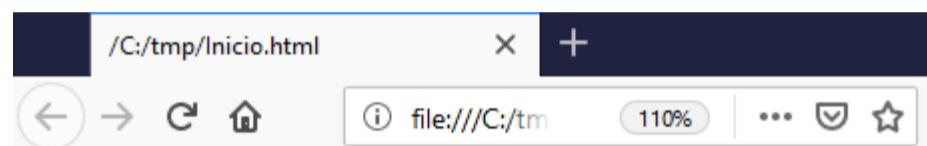


total parámetros: 0
total parámetros: 4
total parámetros: 9
total parámetros: 5

Ilustración 138: Número de parámetros variable en funciones.

```
<!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>
```



```
paso
varios
argumentos
7
1
numeros
3
y texto
```

Ilustración 139: Muestra el contenido de cada parámetro de la función

```
<!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>
```

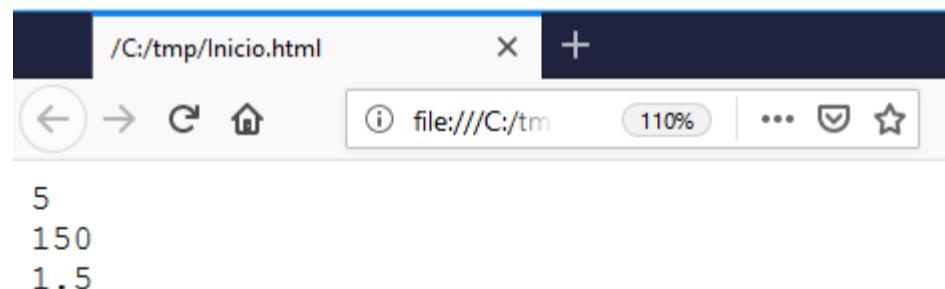


Ilustración 140: Envía una función como parámetro a otra función

```

<!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>

```

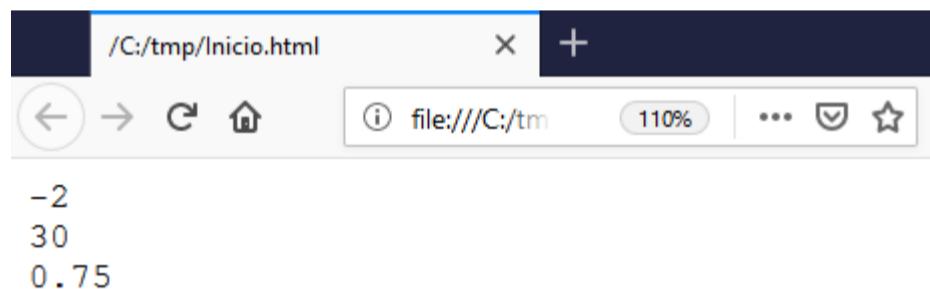


Ilustración 141: Envía una función como parámetro a otra función

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>
```

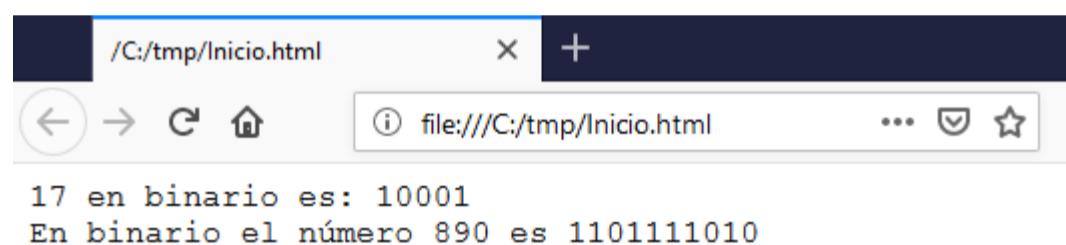


Ilustración 142: Convertir un entero a su representación binaria

```
<!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>
```

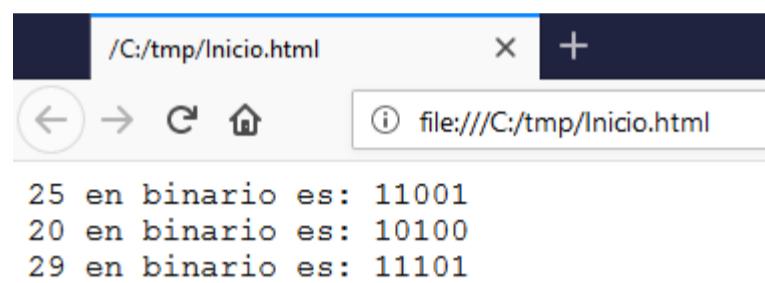


Ilustración 143: Operación OR

```
<!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>
```

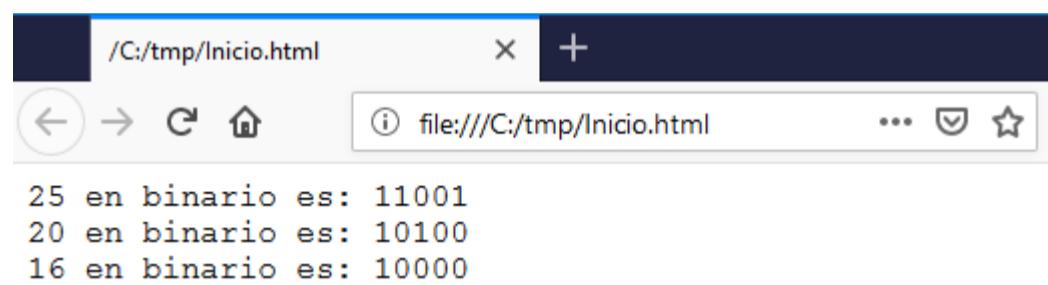


Ilustración 144: Operación AND

```
<!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>
```

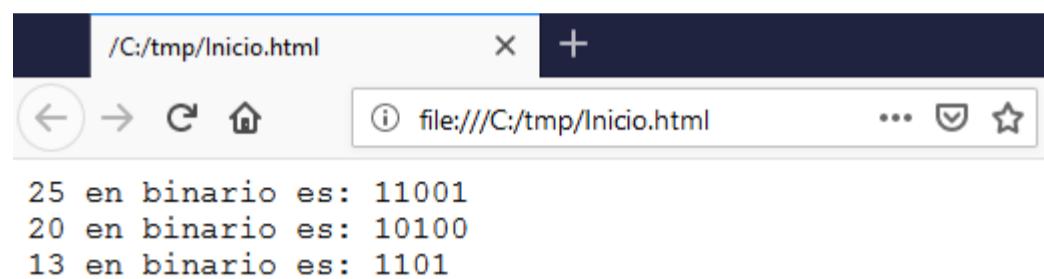


Ilustración 145: Operación XOR

```
<!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>
```

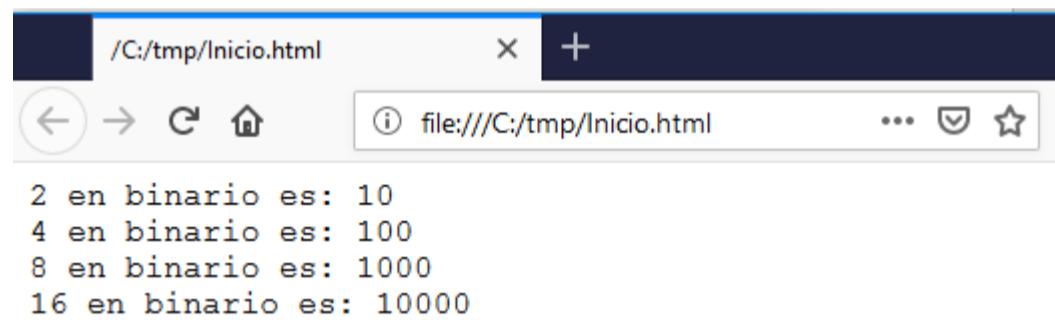


Ilustración 146: Multiplicación usando operaciones de bit

```
<!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>
```

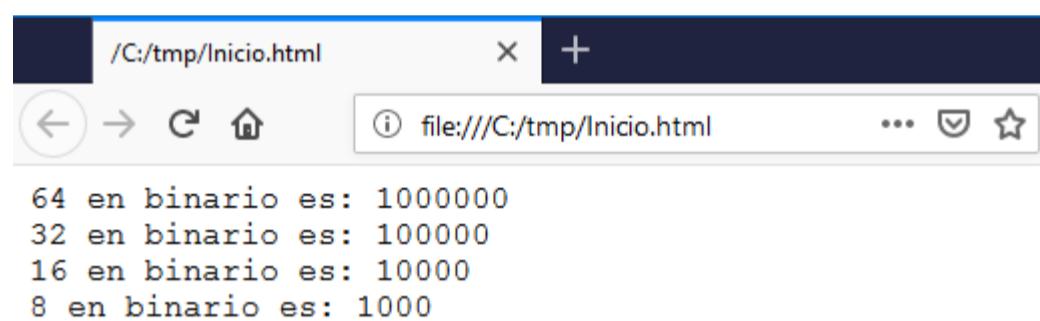


Ilustración 147: División usando operaciones de bit

Intercambiar valores usando operaciones de bit

160.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>
```

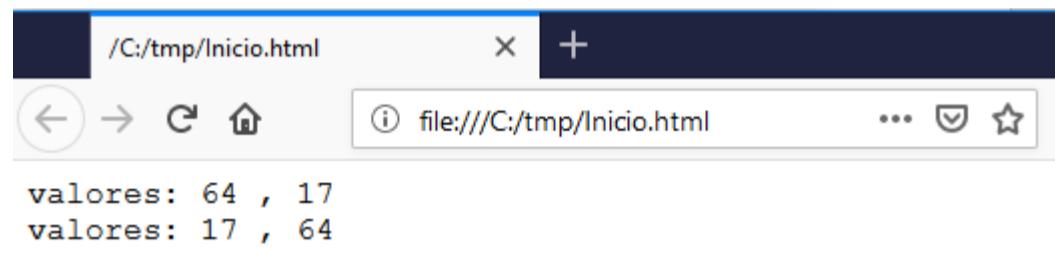


Ilustración 148: Intercambiar valores usando operaciones de bit

Programación orientada a objetos

Se sigue la especificación ECMAScript 2023

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>
```

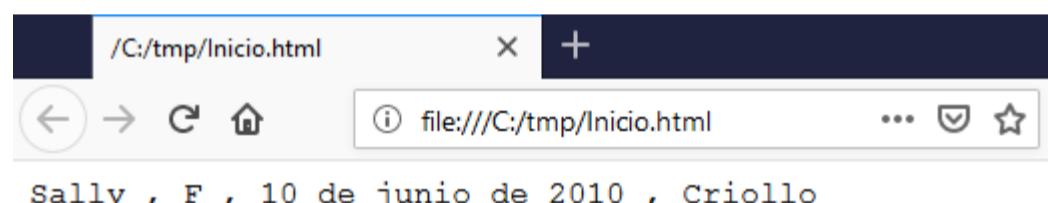


Ilustración 149: Valores de atributos

```
<!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>
```

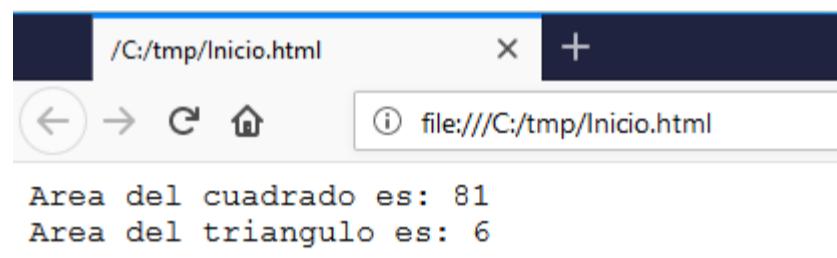


Ilustración 150: Métodos

```
<!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>
```



Ilustración 151: Métodos y atributos

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

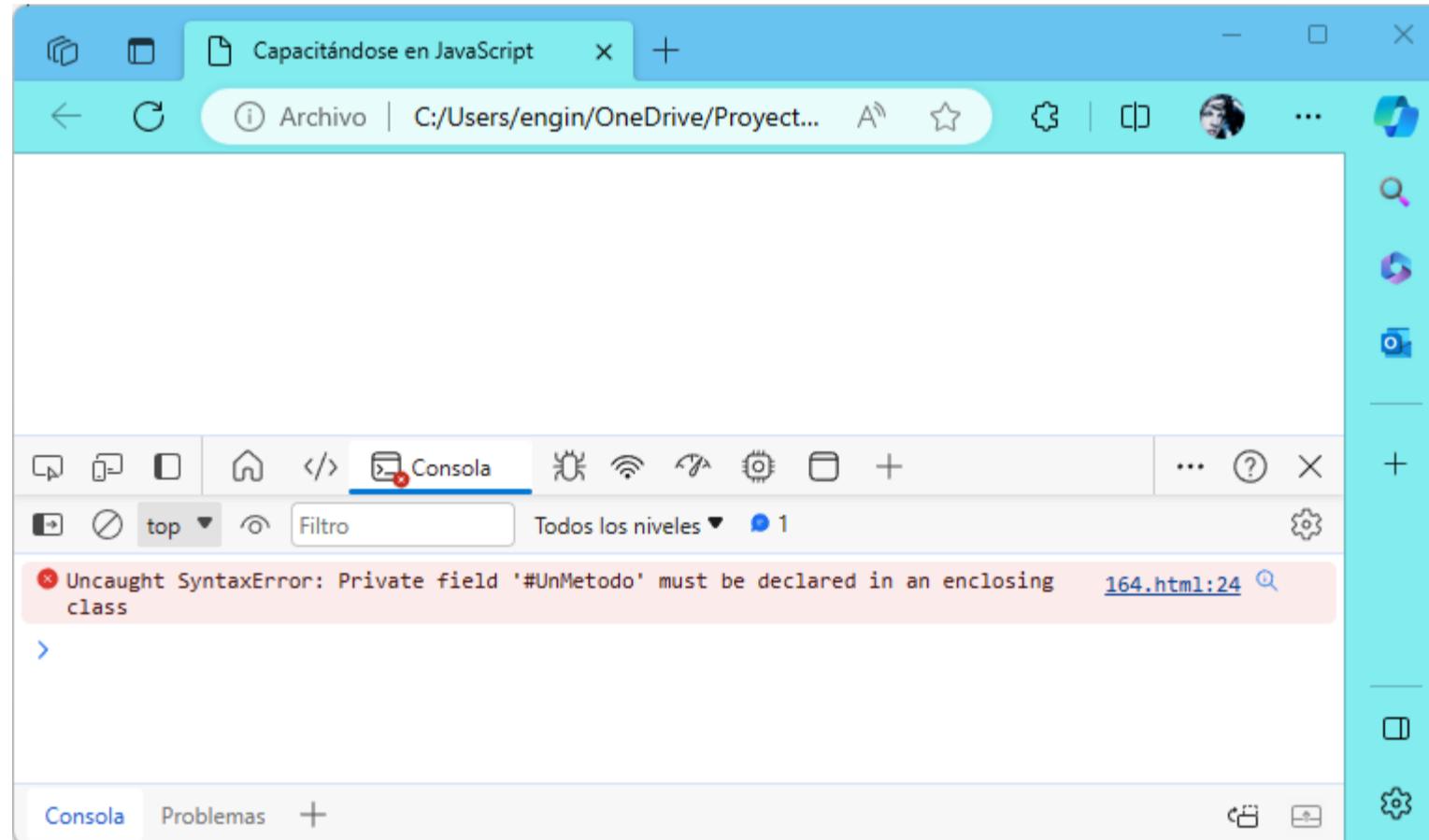


Ilustración 152: Métodos privados

```
<!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>
```

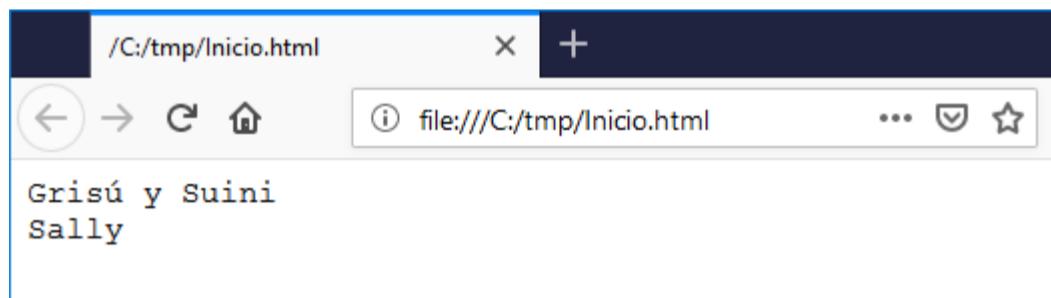


Ilustración 153: Getters y Setters

```
<!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>
```



Ilustración 154: Getters y Setters

```
<!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();

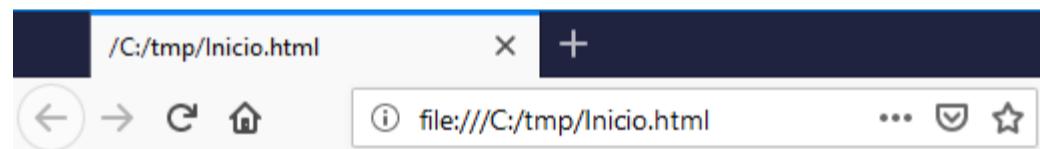
    document.write("<br><br>prueba es una instancia de Hija: ");
    document.write(prueba instanceof Hija); //Resultado true

    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>
```



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

Ilustración 155: Herencia

```
<!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>
```



Ilustración 156: Herencia y constructor

```

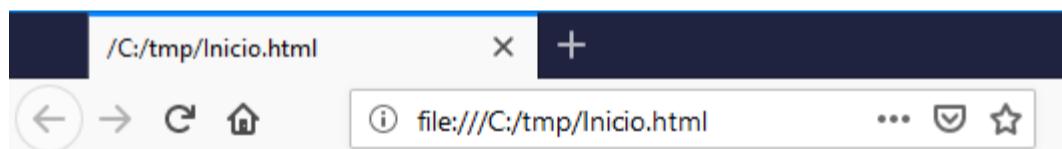
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<script>
    //Clase abuela
    class Abuela {
        constructor() {
            document.write("<br>Constructor clase abuela");
        }
    }

    //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>

```



Constructor clase abuela
 Constructor clase madre
 Constructor clase hija

Ilustración 157: Herencia y constructor

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

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

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>
```

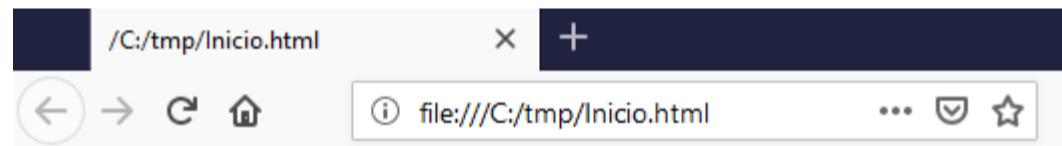


Llama al método A con tres parámetros: undefined, undefined, undefined
Llama al método A con tres parámetros: 2, 7, undefined
Llama al método A con tres parámetros: 8, undefined, undefined
Llama al método A con tres parámetros: 5, 9, 3

Ilustración 158: No hay sobrecarga de métodos en JavaScript

```
<!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>
```



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

Ilustración 159: typeof

```
<!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>
```

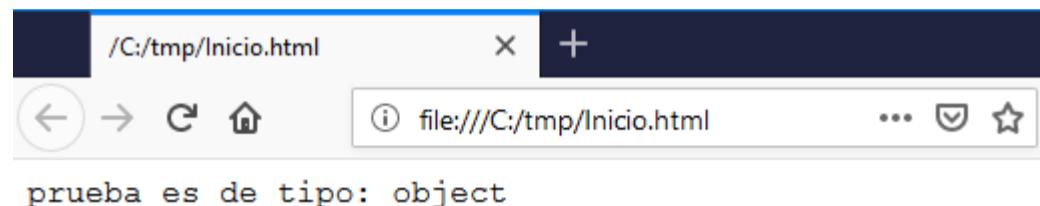


Ilustración 160: typeof de objetos

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>
```

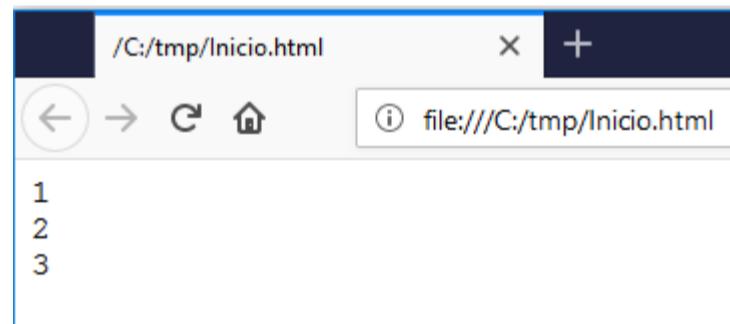


Ilustración 161: Recorrido de árbol binario

Recorrido en pre-orden

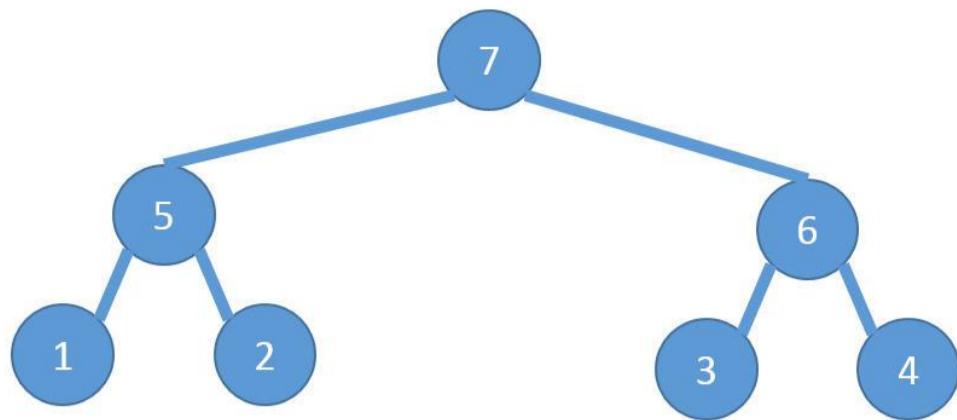


Ilustración 162: Árbol binario

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>
```

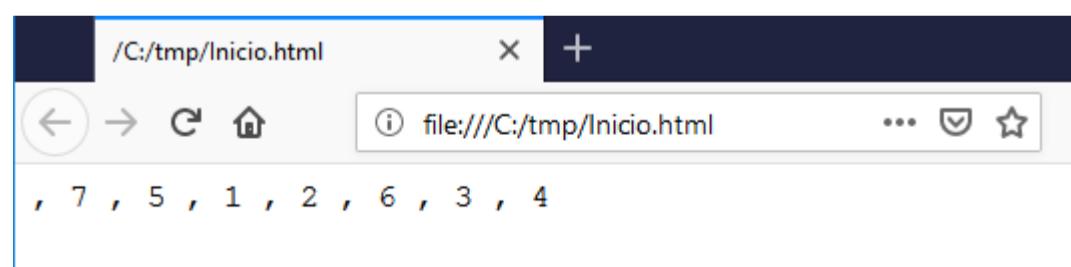


Ilustración 163: Recorrido en pre-orden

```
<!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>
```

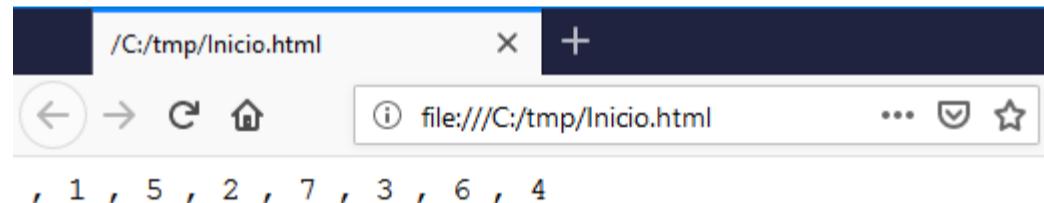


Ilustración 164: Recorrido en in-orden

```
<!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>
```

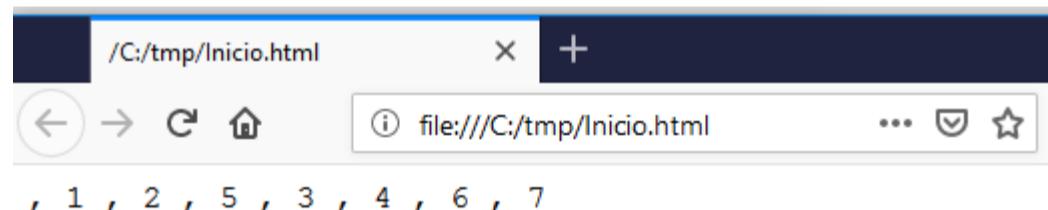


Ilustración 165: Recorrido en post-orden

```

<!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>

```

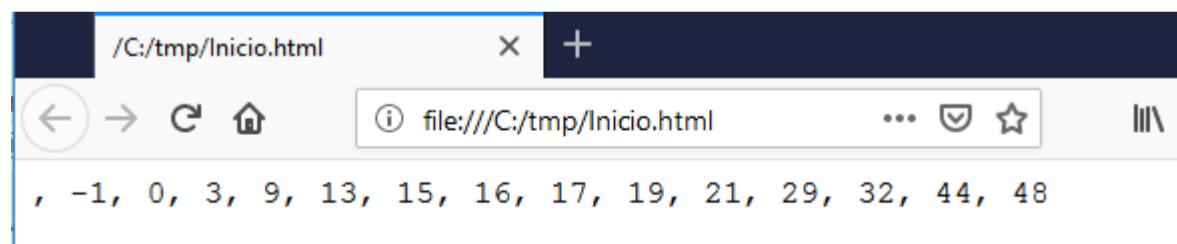


Ilustración 166: Ordenación usando un árbol binario

```
<!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>
```

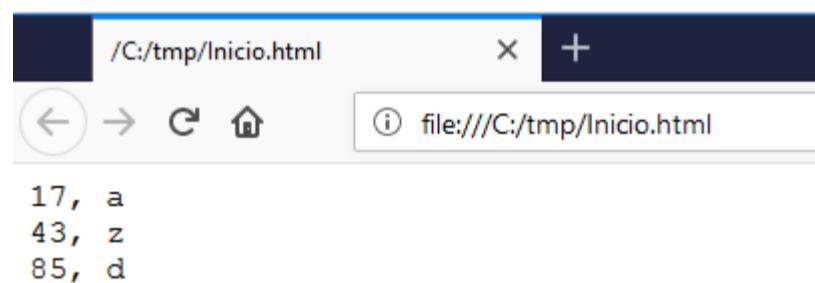


Ilustración 167: Lista de objetos. Pilas.

```

<!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>

```

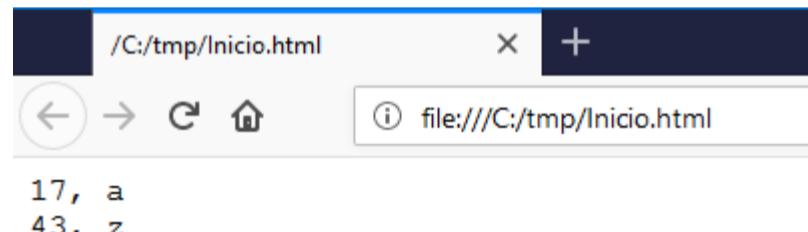


Ilustración 168: Lista de objetos. Pilas.

```
<!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>
```

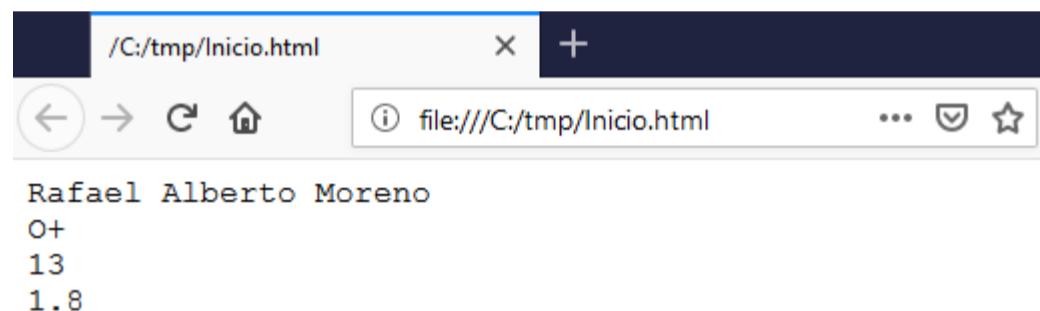


Ilustración 169: Primer ejemplo de uso de JSON

```
<!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>
```

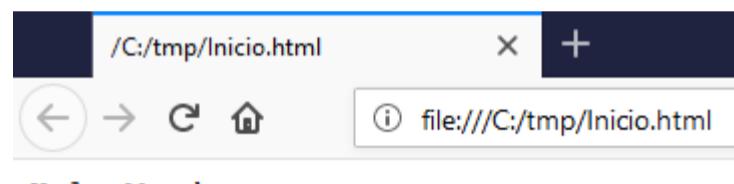


Ilustración 170: Segundo ejemplo de uso de JSON

```
<!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>
```

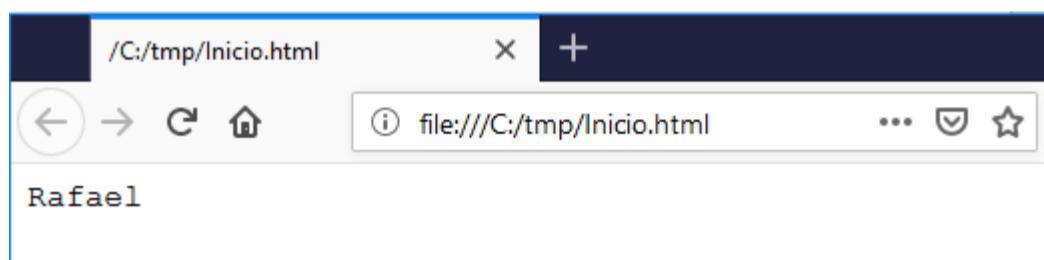


Ilustración 171: Tercer ejemplo de uso de JSON

```
<!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>
```

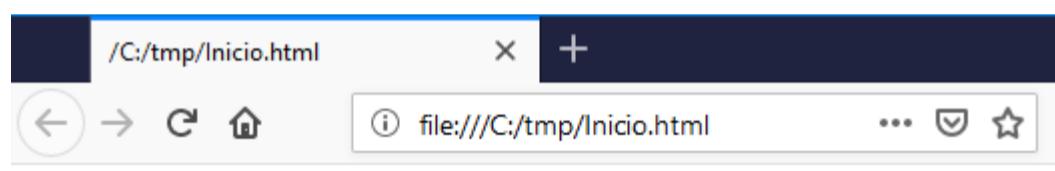


Ilustración 172: Cuarto ejemplo de uso de JSON

```
<!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 Turquí", "esperanzavida": 118}],

        "Reptiles": [
            {"nombre": "Tortuga radiada", "esperanzavida": 100},
            {"nombre": "Cocodrilo", "esperanzavida": 50},
            {"nombre": "Dragón de Komodo", "esperanzavida": 30}]
    };
    document.write(animal.Aves[0].nombre + "<br>");
    document.write(animal.Reptiles[1].nombre);

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

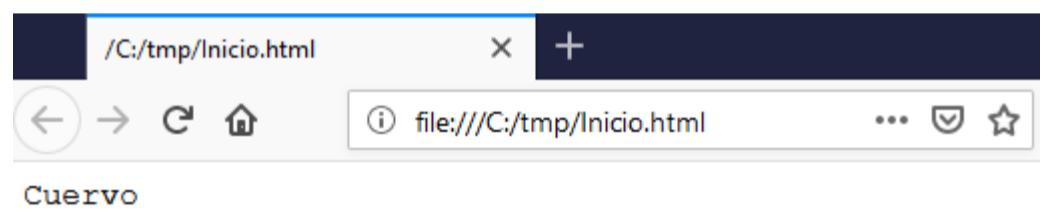


Ilustración 173: Quinto ejemplo de uso de JSON

```

<!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>

```

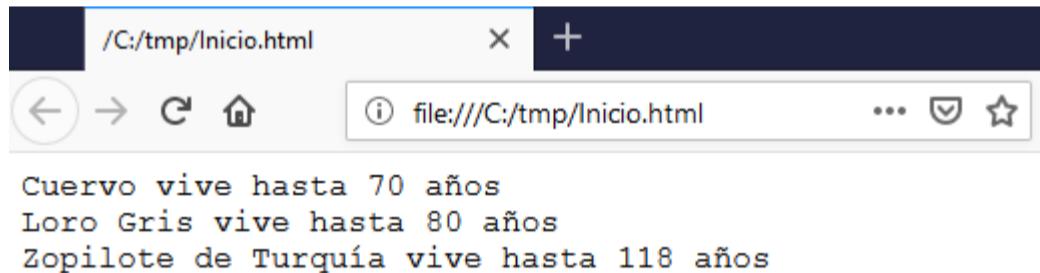


Ilustración 174: Sexto ejemplo de uso de JSON

```

<!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>

```

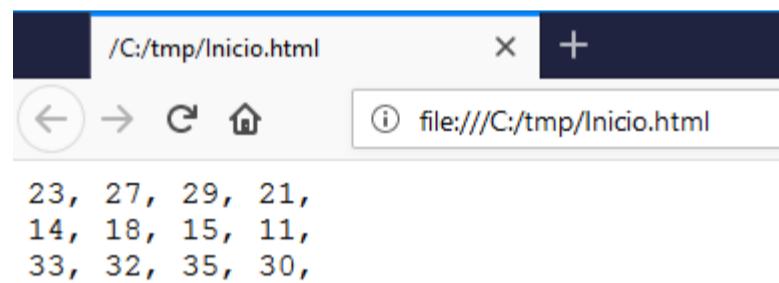


Ilustración 175: Séptimo ejemplo de uso de JSON

```

<!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", "kqxrst"]
        }
    };

    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,kqxrst
    xyzwqr
    */
</script>
</body>
</html>

```

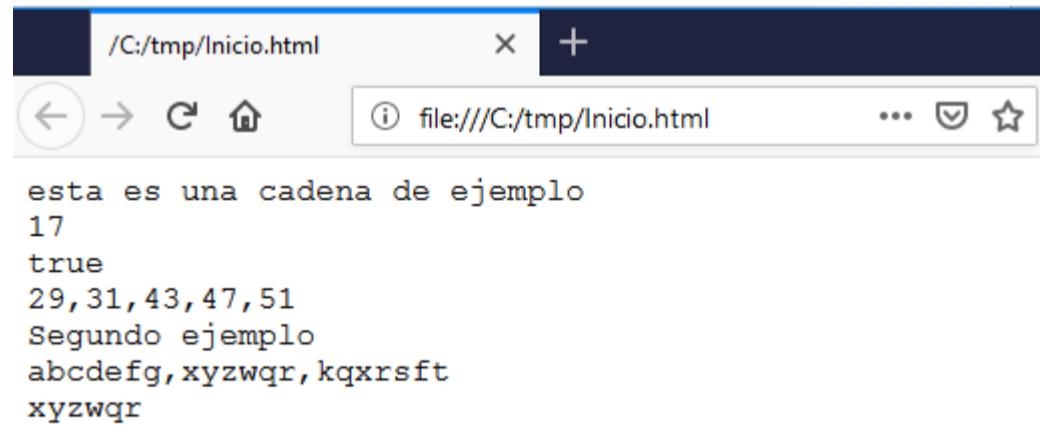


Ilustración 176: Octavo ejemplo de uso de JSON

Control sobre la página HTML

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

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

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>
```

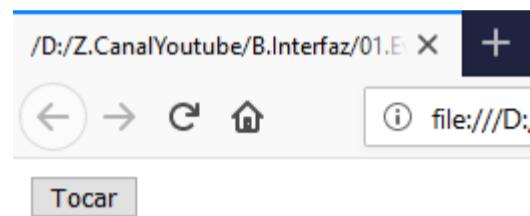


Ilustración 177: Botón

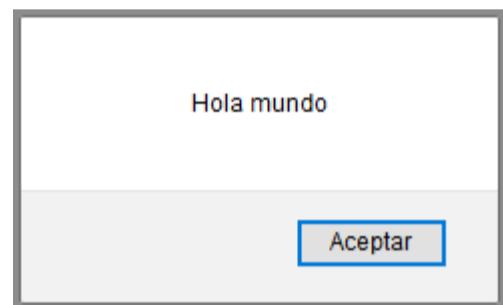


Ilustración 178: Ventana emergente generada por JavaScript

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>
```

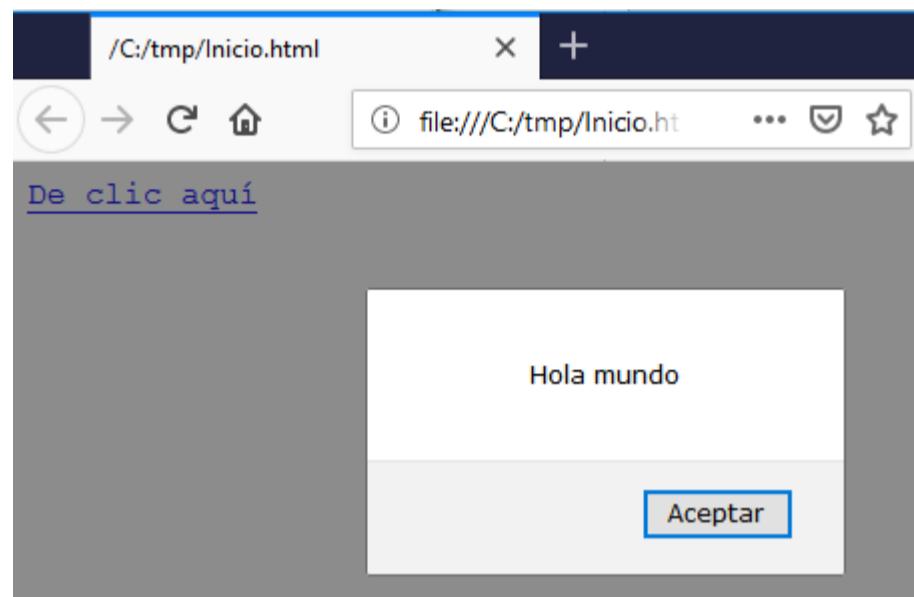


Ilustración 179: Ventana emergente al dar clic en un hipervínculo

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>
```

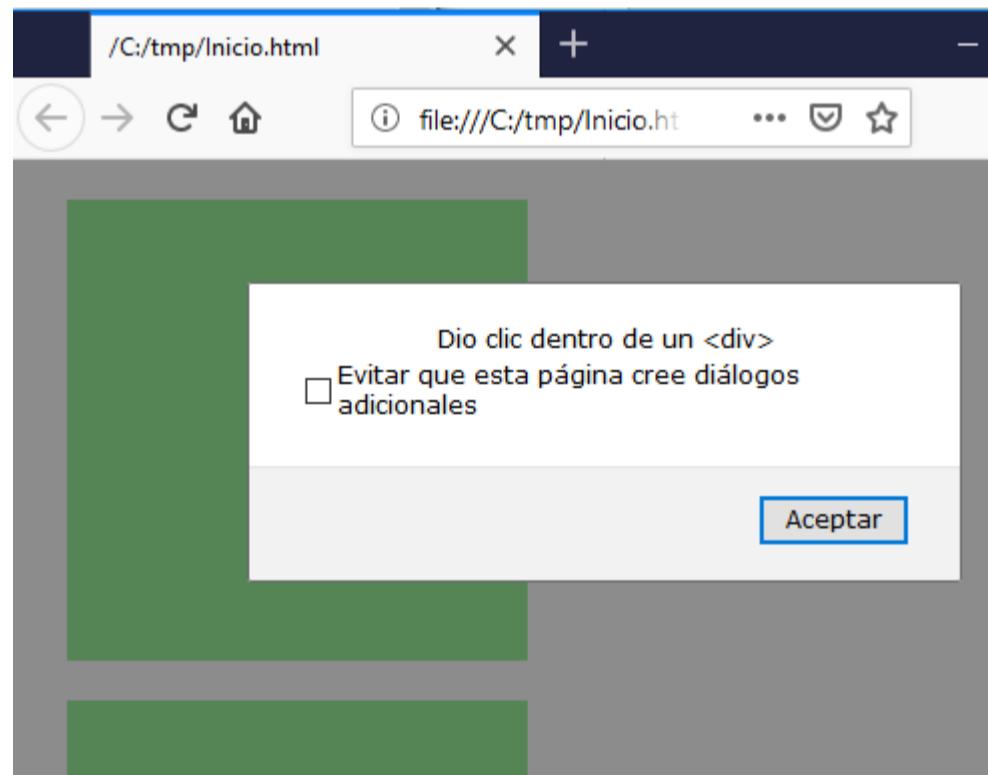


Ilustración 180: Ventana emergente al dar clic en un <div>

```
<!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>
```

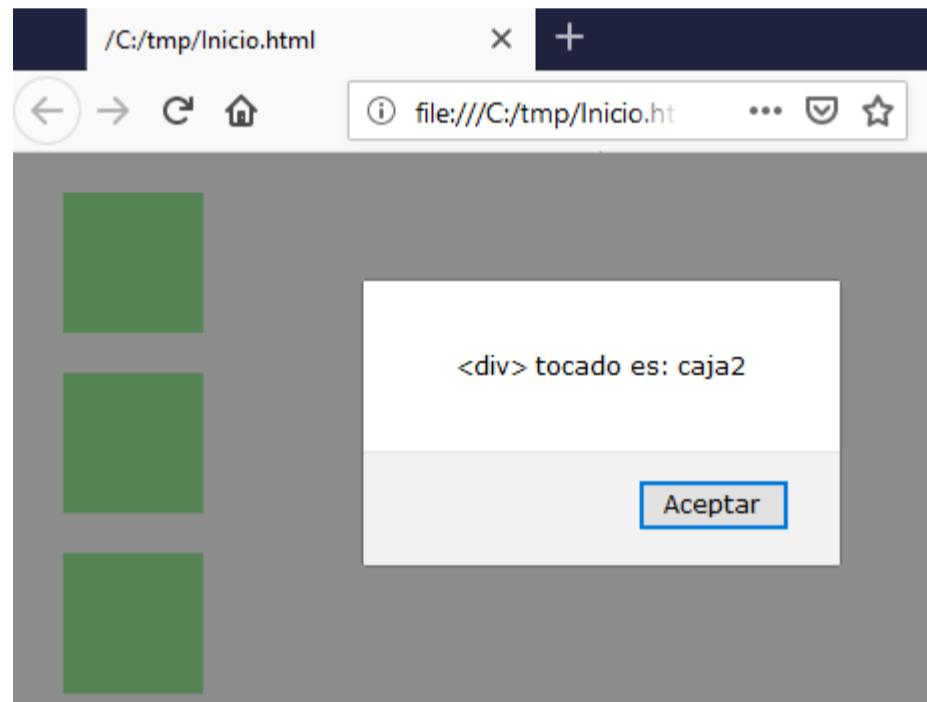


Ilustración 181: Ventana emergente al dar clic en un <div>

```
<!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 divdobleclick() {
        alert("dio doble clic");
    }
</script>
<div id="caja1" ondblclick="divdobleclick()"></div>
</body>
</html>
```

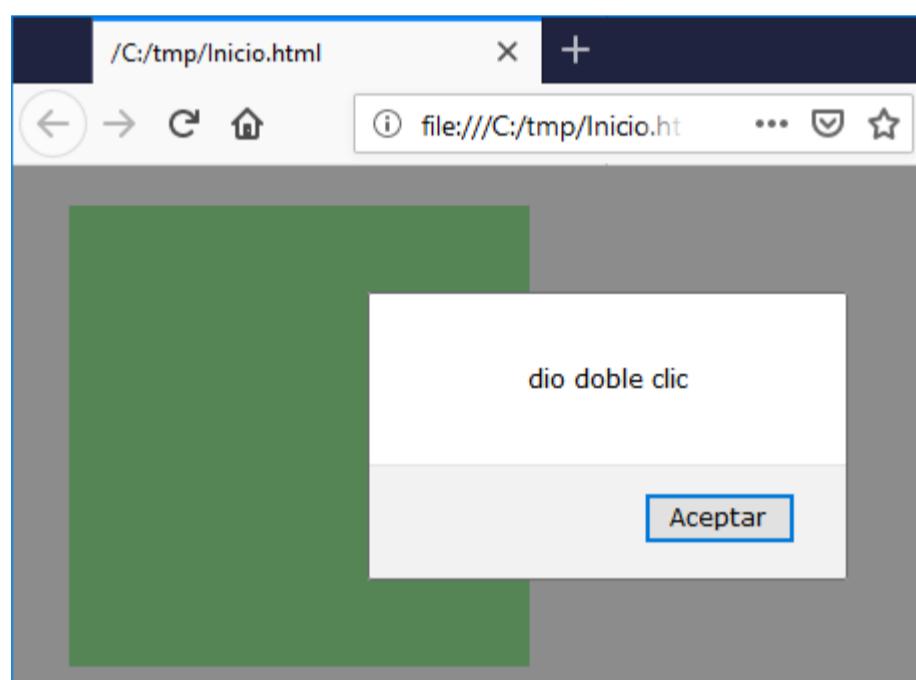


Ilustración 182: Ventana emergente al dar doble clic en un <div>

```
<!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>
```

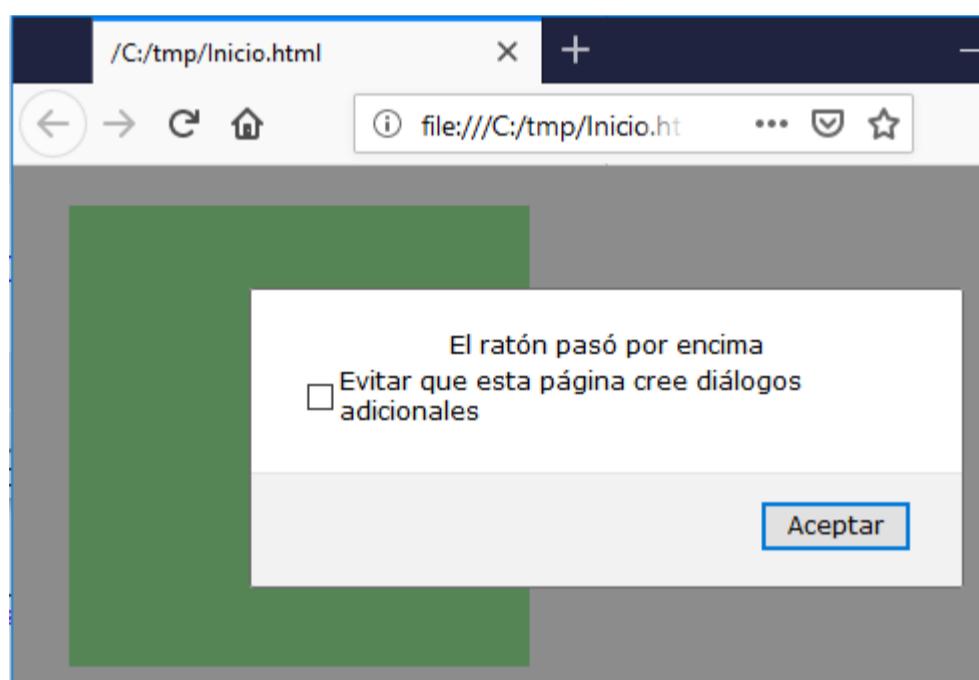


Ilustración 183: Ventana emergente al mover el cursor por encima de un <div>

```
<!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>
```

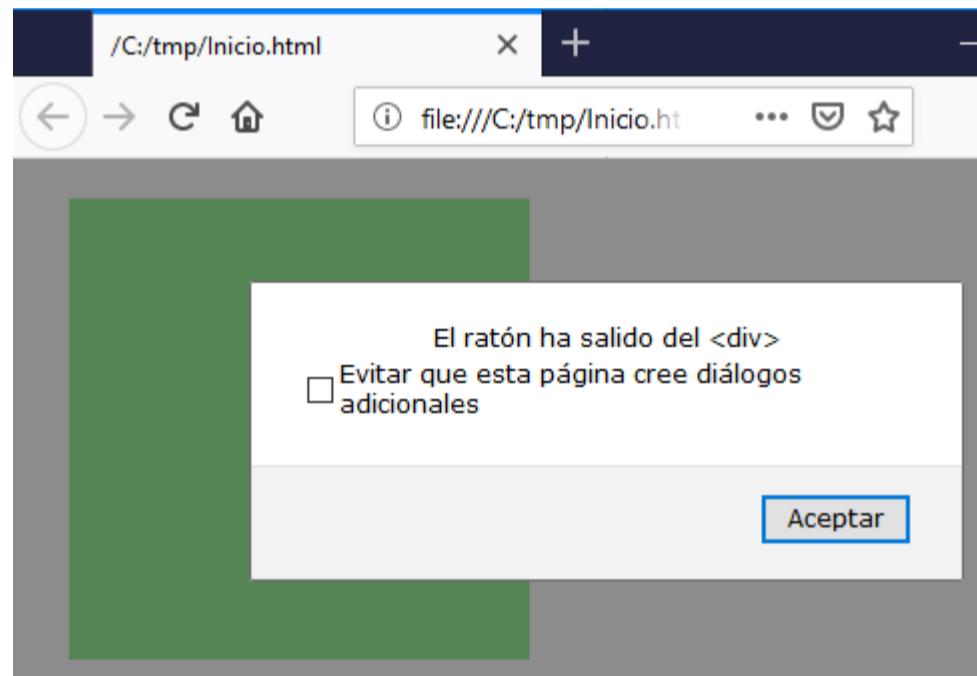


Ilustración 184: Ventana emergente al mover el cursor fuera de un <div>

```
<!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>
```

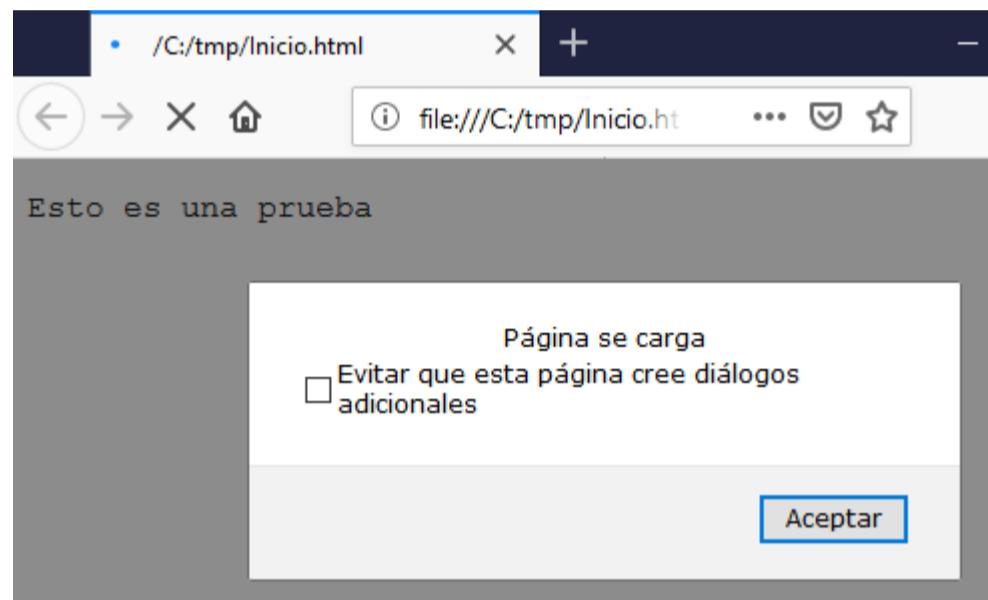


Ilustración 185: Ventana emergente al entrar a una página

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

205.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>
```

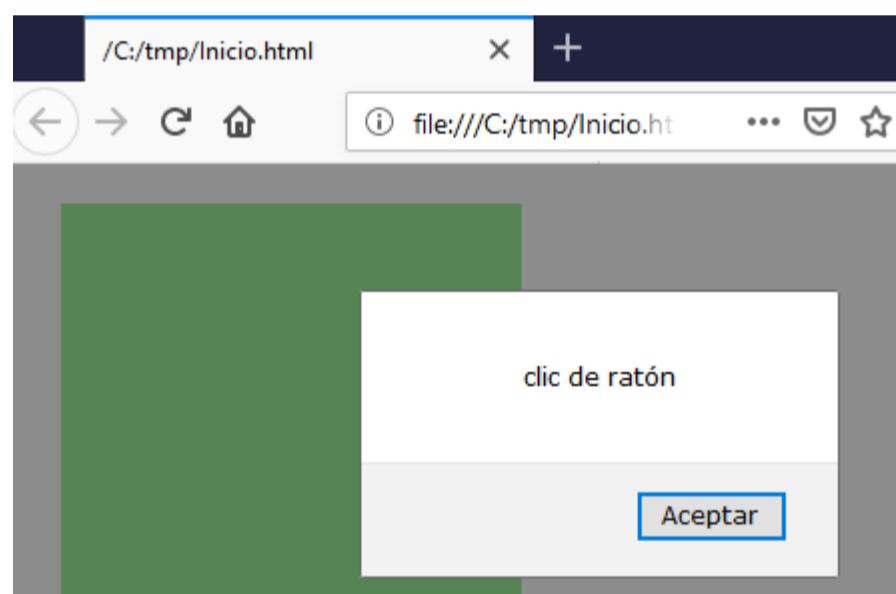


Ilustración 186: Dar clic al interior de un <div>

```
<!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>
```

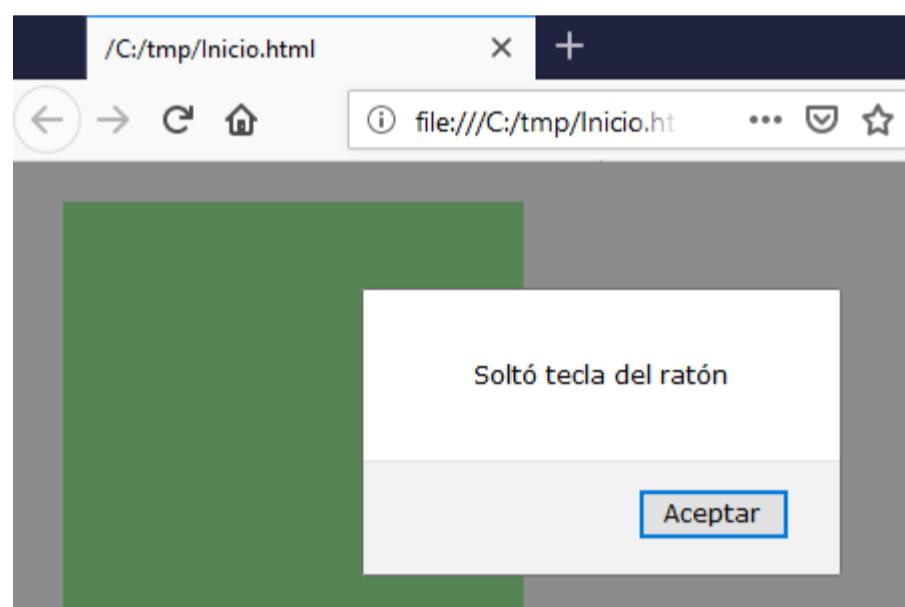


Ilustración 187: Soltar la tecla izquierda del ratón en el interior de un <div>

Captura el evento de mover el cursor del ratón

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

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>
```



Ilustración 188: Cambiar aleatoriamente el color de fondo al mover el cursor del ratón al interior de un <div>

Captura el evento cuando se cambia el tamaño de la ventana

208.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>
```

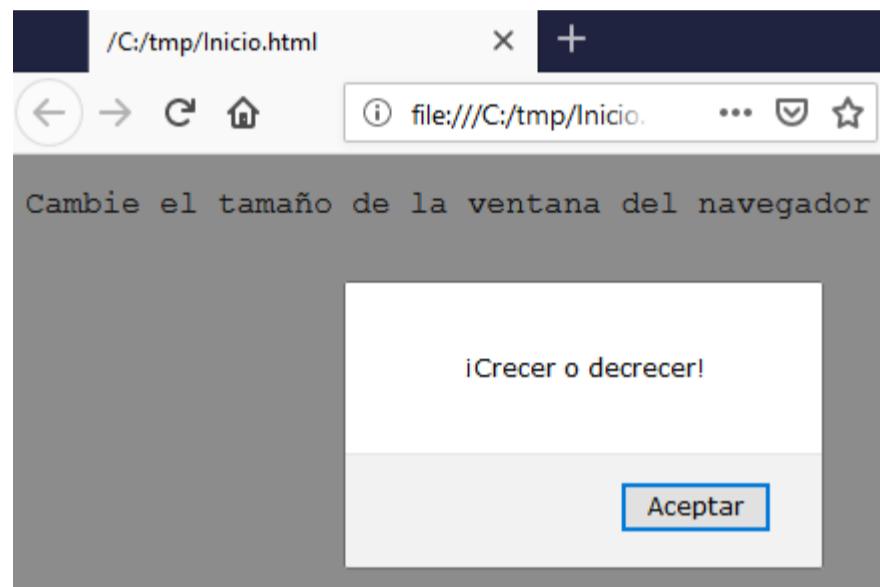


Ilustración 189: Mostrar un aviso cuando se cambia el tamaño de la ventana del navegador

```
<!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>
```

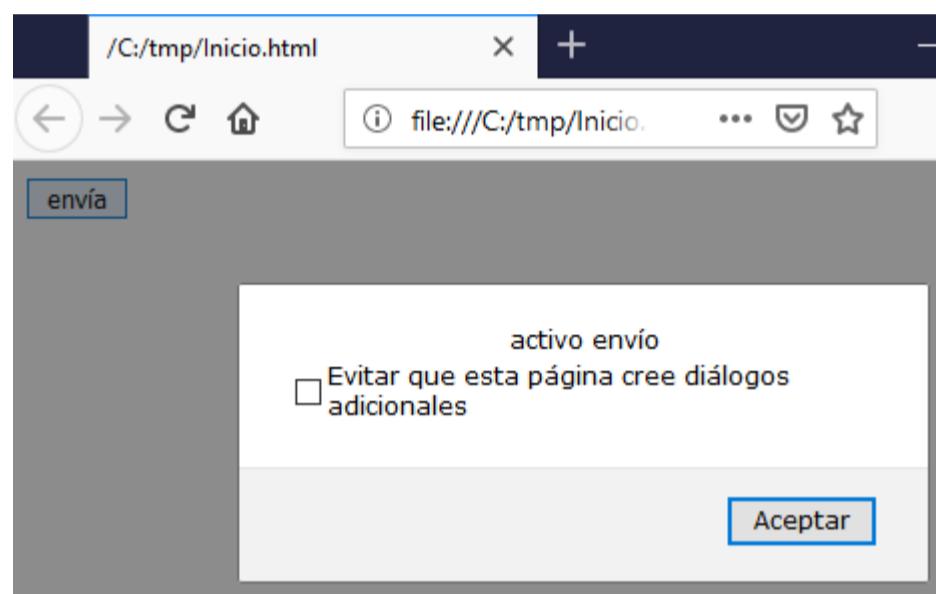


Ilustración 190: Mensaje cuando se presiona botón de envío de formulario

```
<!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>
```

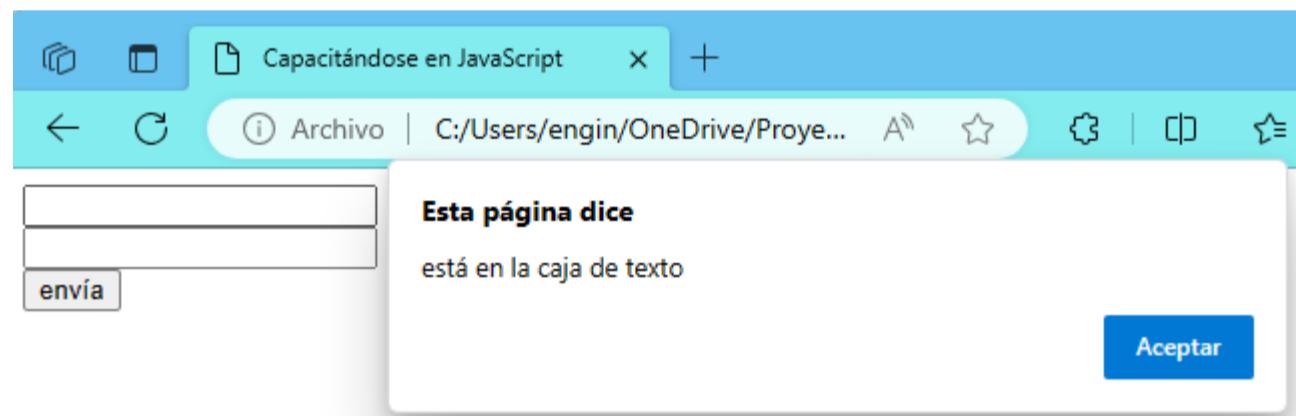


Ilustración 191: Mensaje cuando accede a una caja de texto

```
<!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>
```

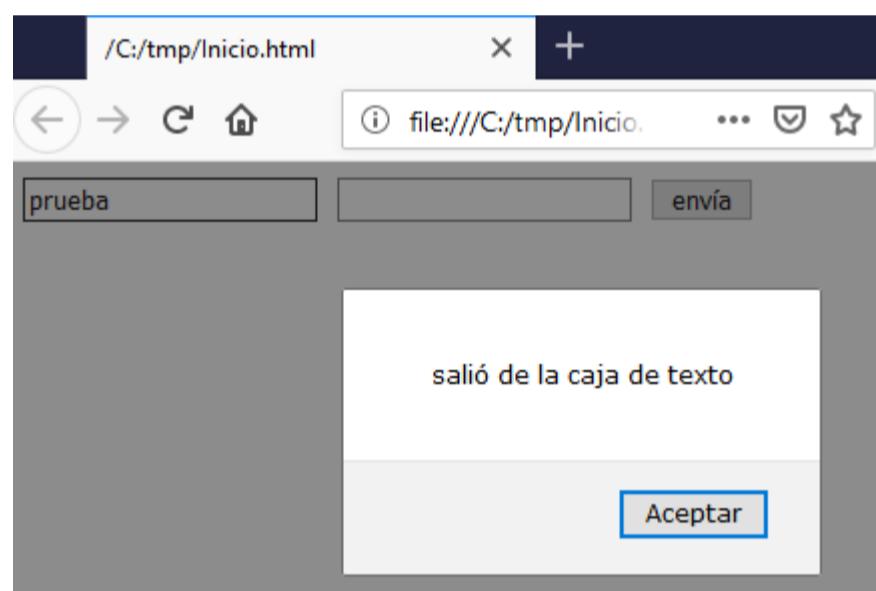


Ilustración 192: Mensaje cuando se sale de una caja de texto

```
<!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>
```

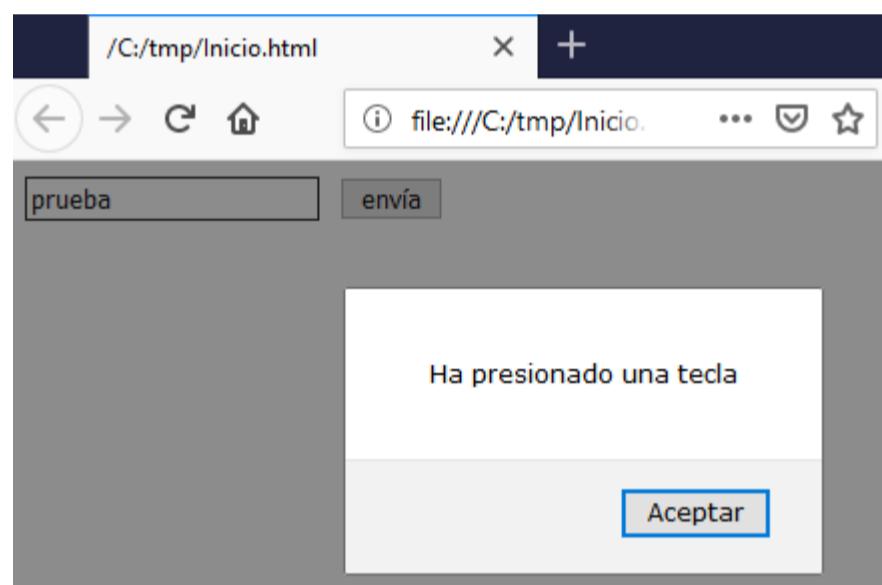


Ilustración 193: Mensaje cuando se presiona cualquier tecla dentro de la caja de texto

Un efecto similar sucede con “onkeypress”

```
<!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>
```

```
<!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="suelteclaela()"/>
    <input type="submit" id="boton" value="envía"/>
</form>
</body>
</html>
```

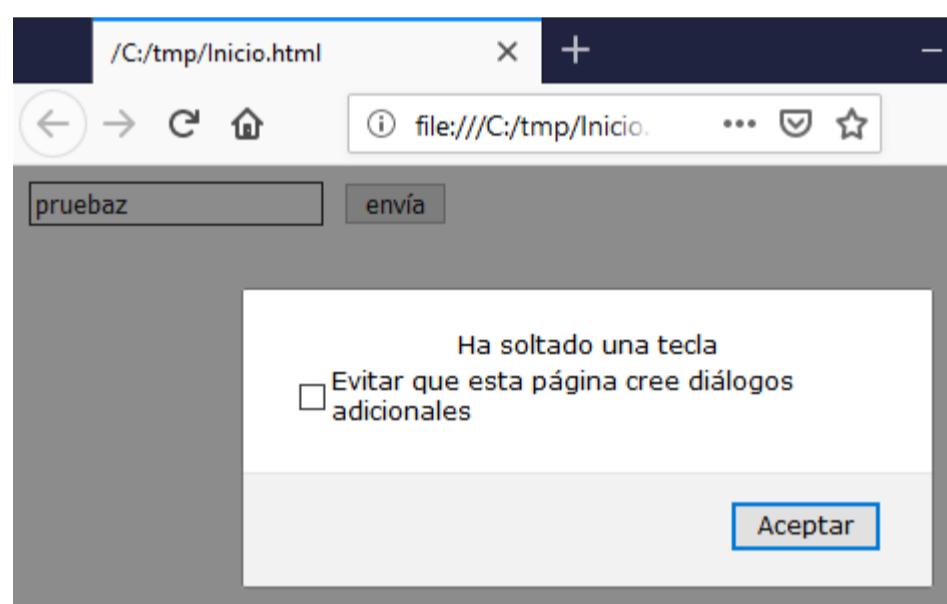


Ilustración 194: Mensaje cuando deja de presionar una tecla en una caja de texto

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

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

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>
```

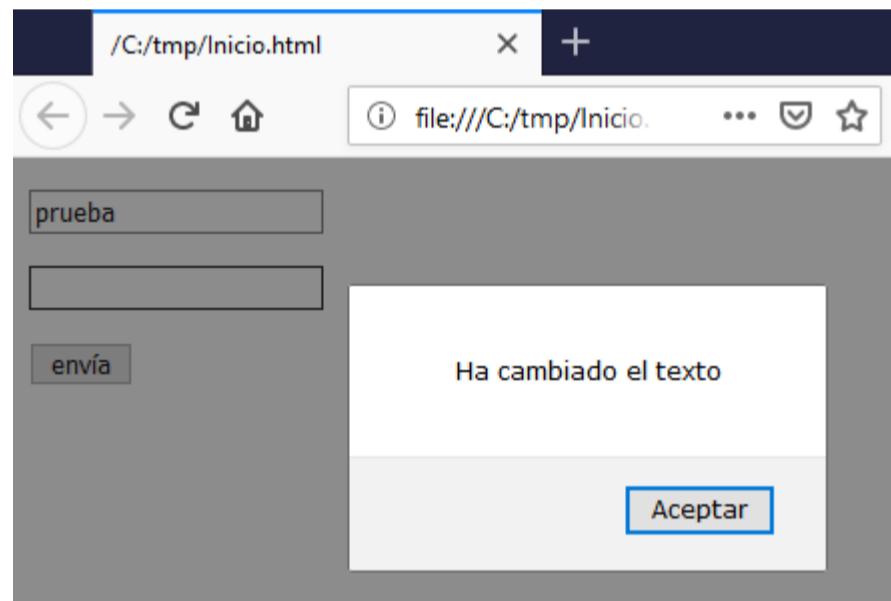


Ilustración 195: Mensaje cuando se ha hecho un cambio dentro de una caja de texto

```
<!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>
```

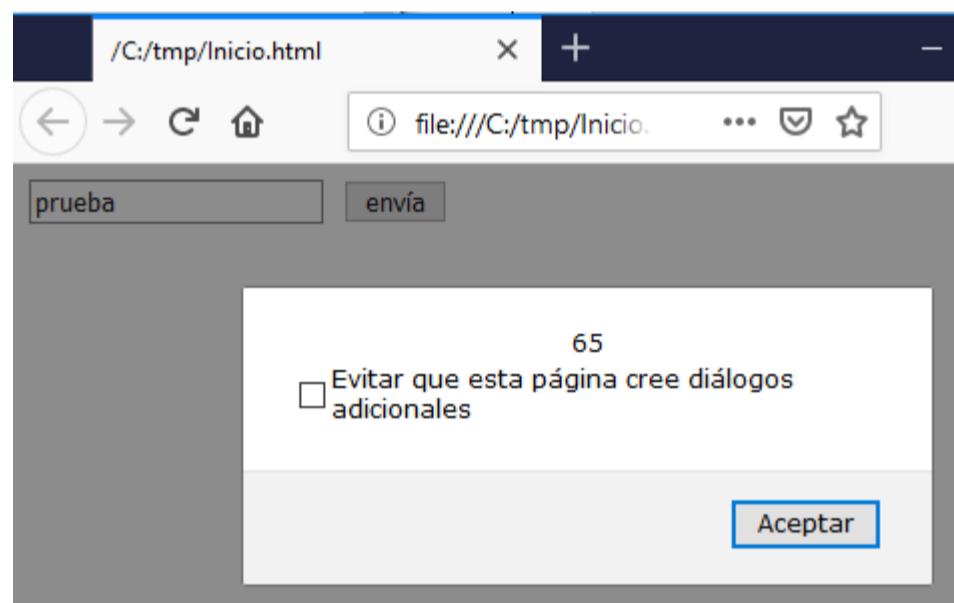


Ilustración 196: Mostrar el código ASCII de la tecla presionada

```
<!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>
```

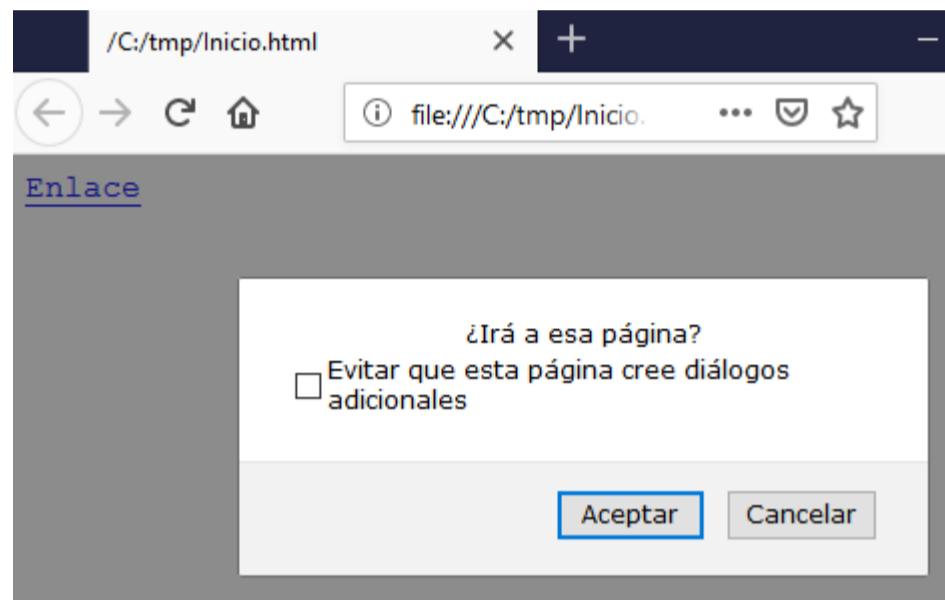


Ilustración 197: Intercepta el poder navegar a otra página

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>
```

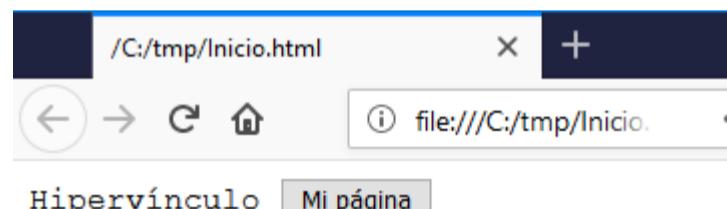


Ilustración 198: Una etiqueta común

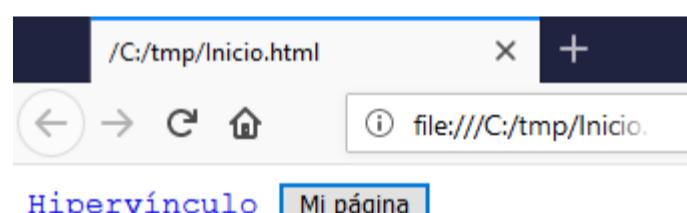


Ilustración 199: La etiqueta común se convierte en hipervínculo

```
<!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>
```

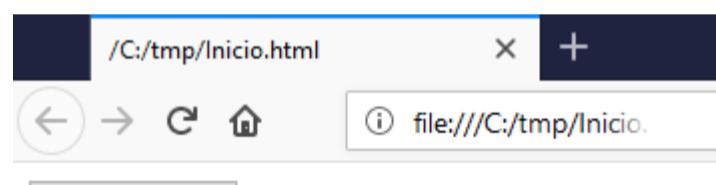


Ilustración 200: Un botón

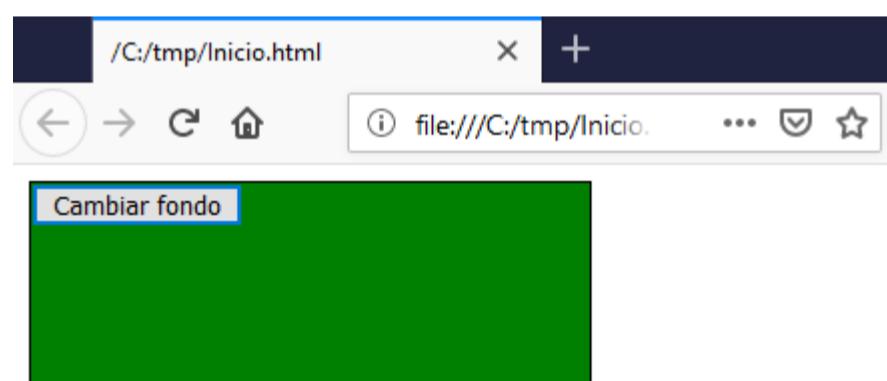


Ilustración 201: Se cambia el color de fondo de un <div> al presionar el botón

```
<!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>
```



Ilustración 202: Un <div> y dos botones

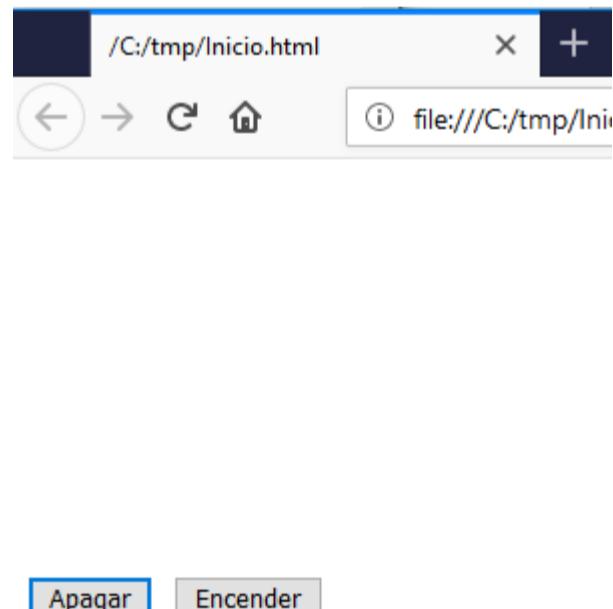


Ilustración 203: Un botón vuelve invisible el <div>

```

<!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>

```



Ilustración 204: Un <div> con texto y dos botones

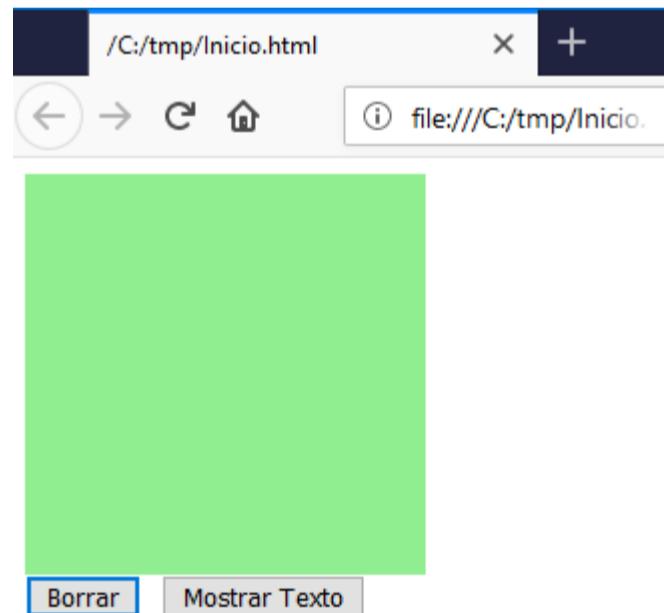


Ilustración 205: Al presionar el botón de "Borrar" se borra el texto del <div>

```
<!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>
```

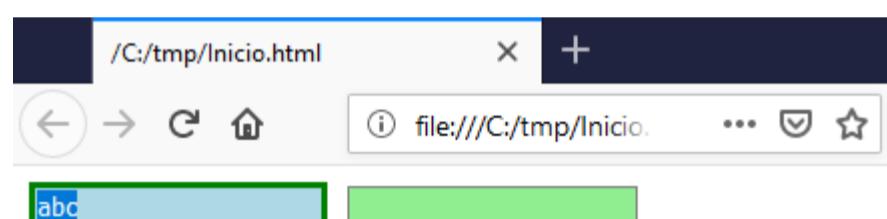


Ilustración 206: Cuando la caja de texto está seleccionada tiene un estilo

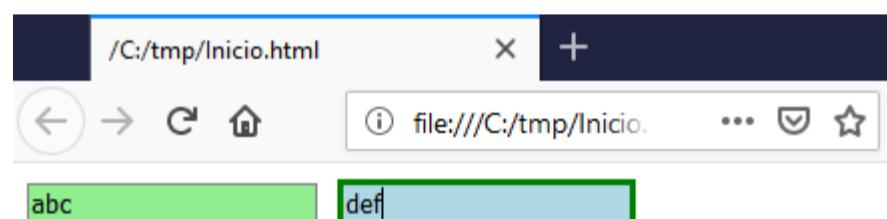


Ilustración 207: Cuando la caja de texto pierde el foco, tendrá otro estilo

```
<!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>

```

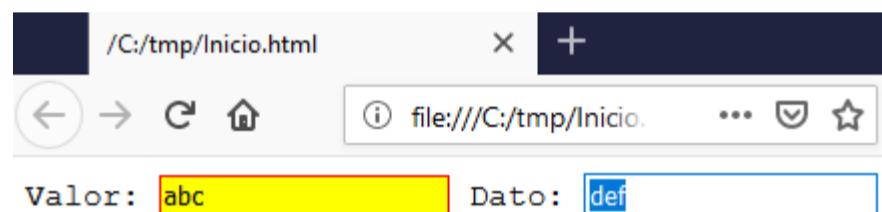


Ilustración 208: Muestra con determinado color si el dato ingresado no es un número válido

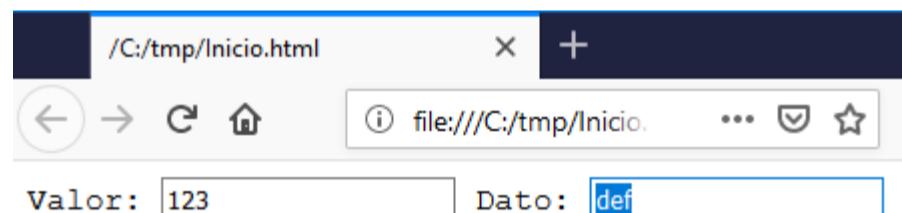


Ilustración 209: Si es un número válido, entonces deja el color por defecto

```
<!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>
```

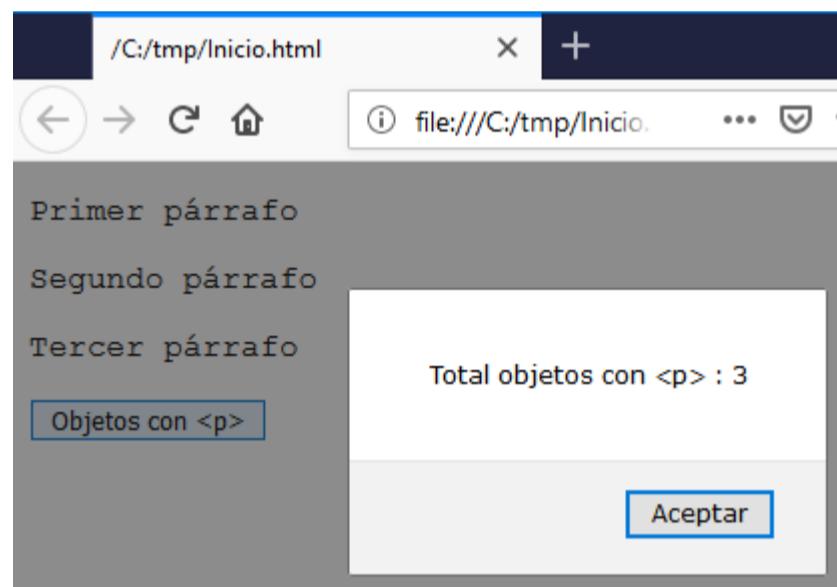


Ilustración 210: Cuenta los objetos con la etiqueta <p>

```
<!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>
```

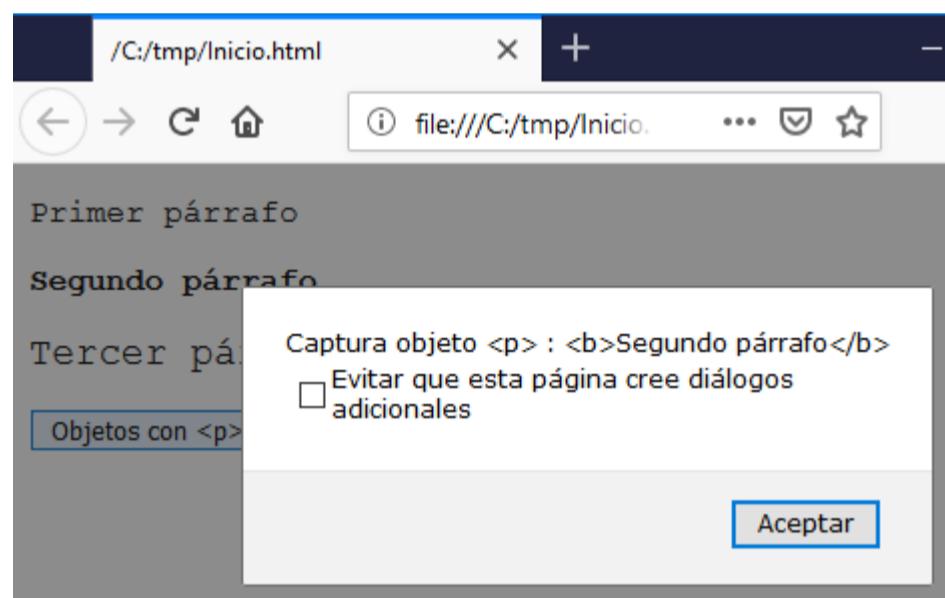


Ilustración 211: Captura un determinado objeto con <p> y lo examina

```
<!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>
```

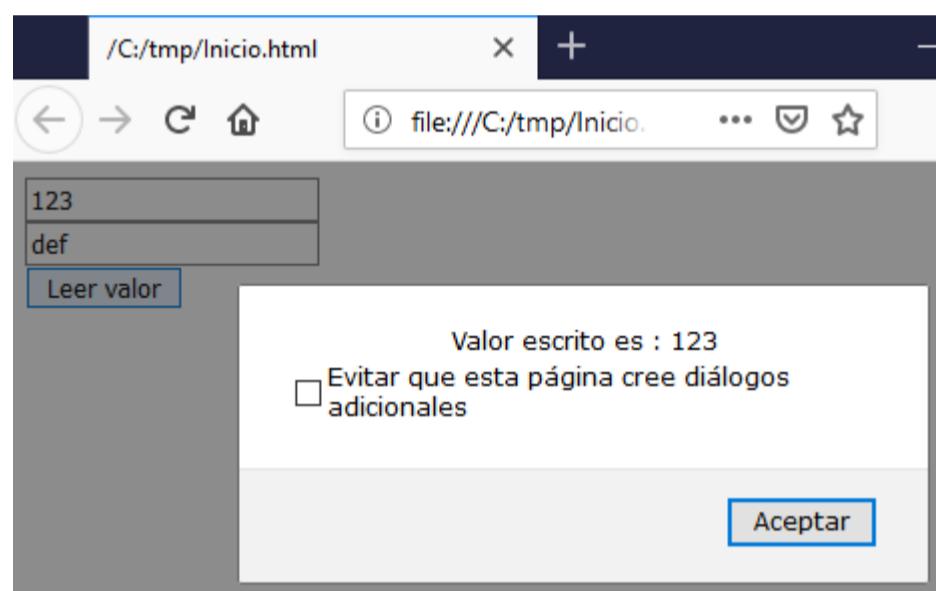


Ilustración 212: Obtiene la lista de objetos tipo entrada, trae el primero y muestra su valor

```
<!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>
```

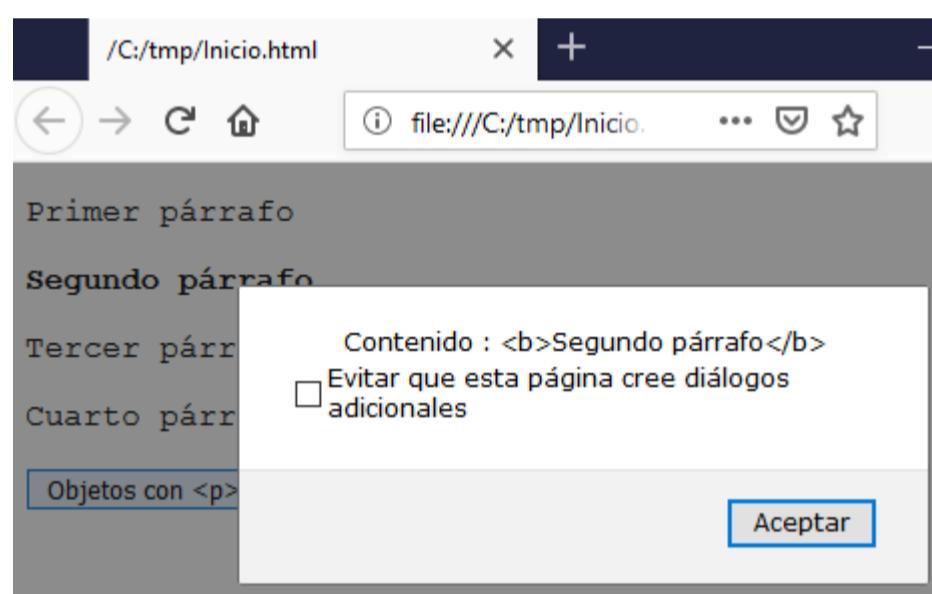


Ilustración 213: Muestra el valor que tiene cada objeto <p>

```

<!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>

```

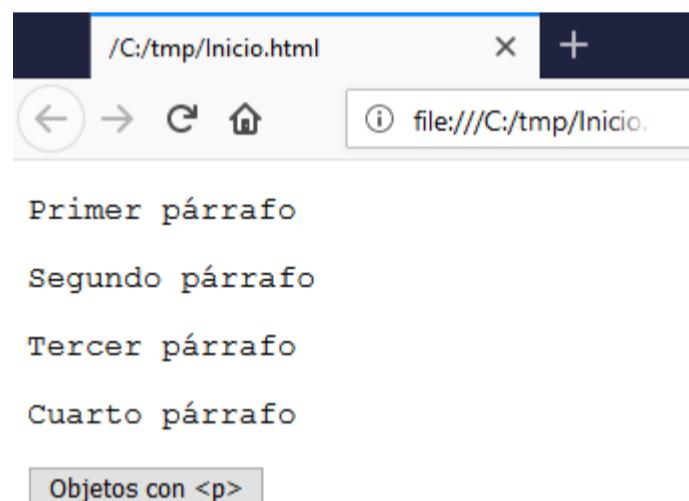
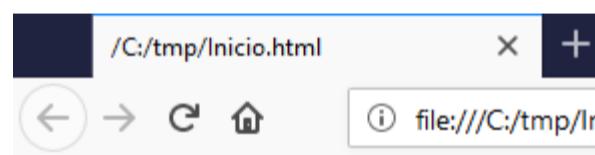


Ilustración 214: Párrafos <p> con estilo por defecto



Segundo párrafo

Tercer párrafo

Cuarto párrafo

Objetos con <p>

Ilustración 215: Se cambia el estilo de cada objeto <p>

```
<!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="¿Donde estoy?">
</body>
</html>
```

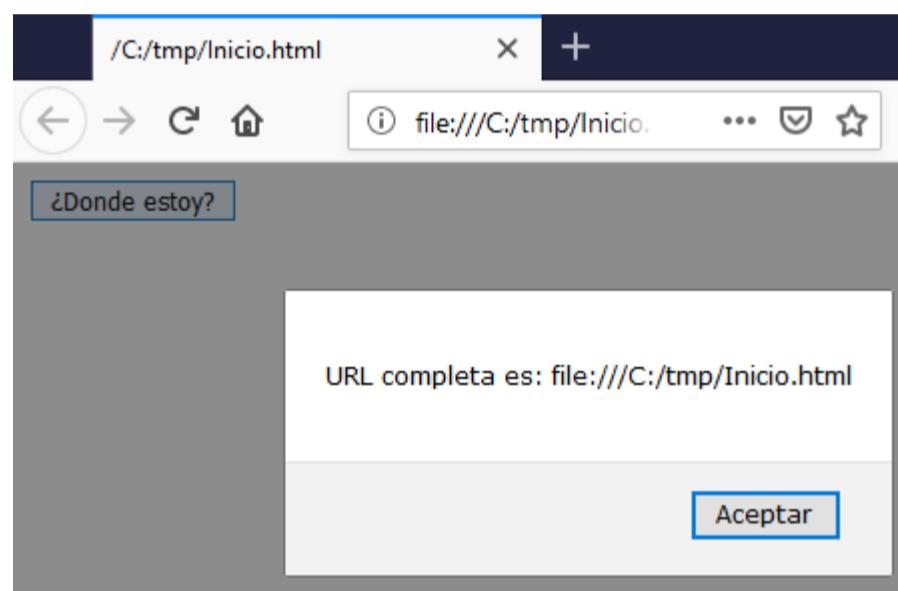
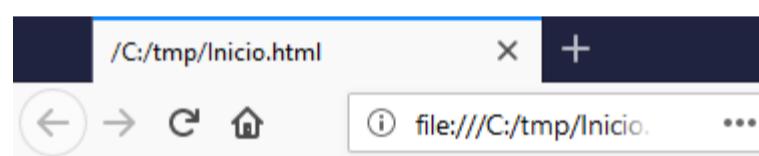


Ilustración 216: Muestra la URL donde se encuentra la página

```
<!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>
```

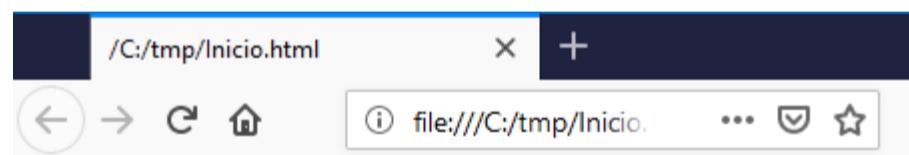


Esta es una prueba

Segundo texto

Aplicar Atributo

Ilustración 217: Dos objetos <p>



Esta es una prueba

Segundo texto

Aplicar Atributo

Ilustración 218: Se pueden cambiar los atributos de un objeto <p> en particular

```
<!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>
```

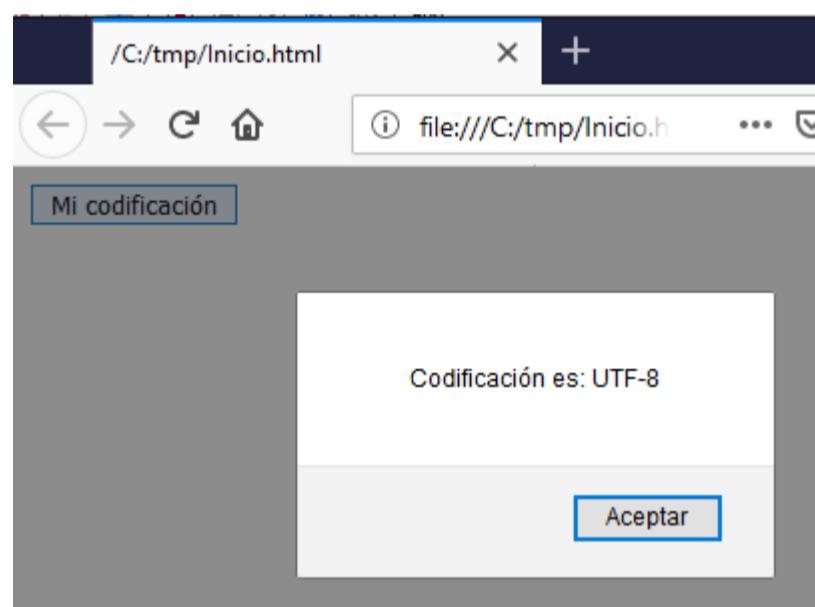


Ilustración 219: Muestra la codificación que tiene la página

```
<!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>
```

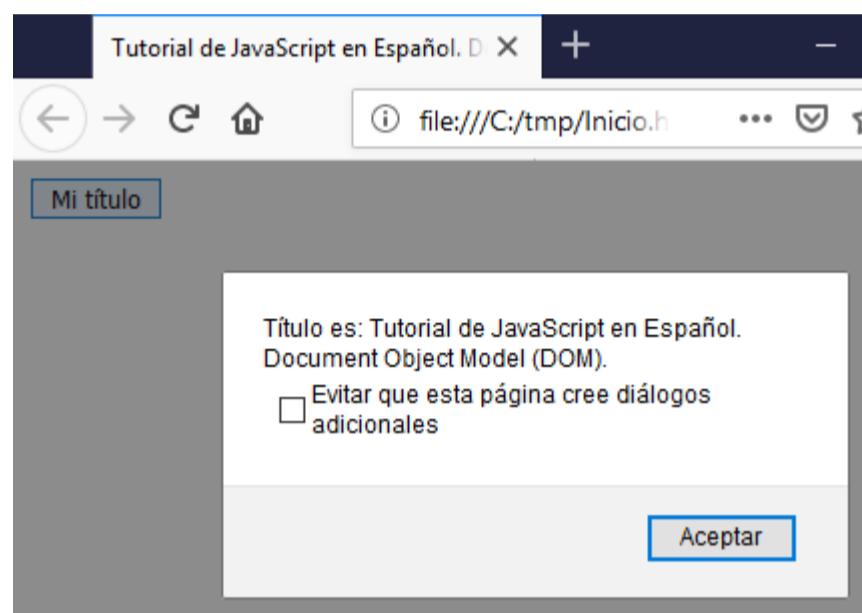
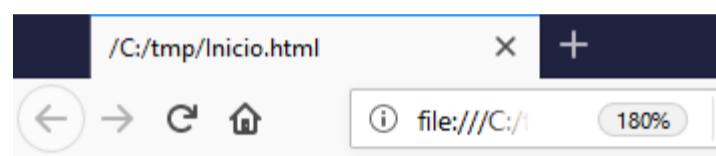


Ilustración 220: Muestra el título que tiene la página

```
<!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>
```



Esta es una prueba
de escritura de texto
en diferentes líneas

Ilustración 221: Saltos de línea sin usar


```
<!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>
```

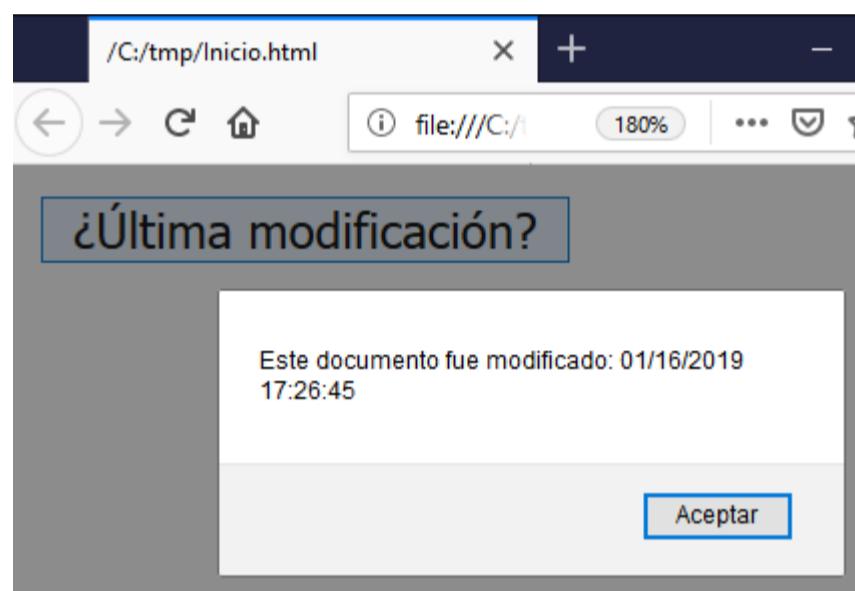


Ilustración 222: Muestra las fechas de modificación de la página

```
<!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>
```

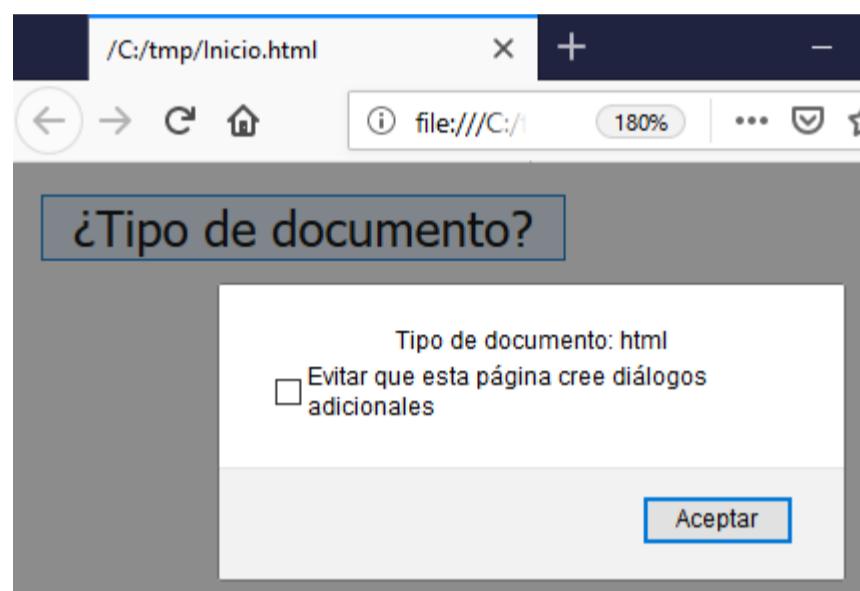


Ilustración 223: Muestra el tipo de documento

```
<!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>
```



Ilustración 224: Mostrar el dominio donde está alojada la página

```
<!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>
```

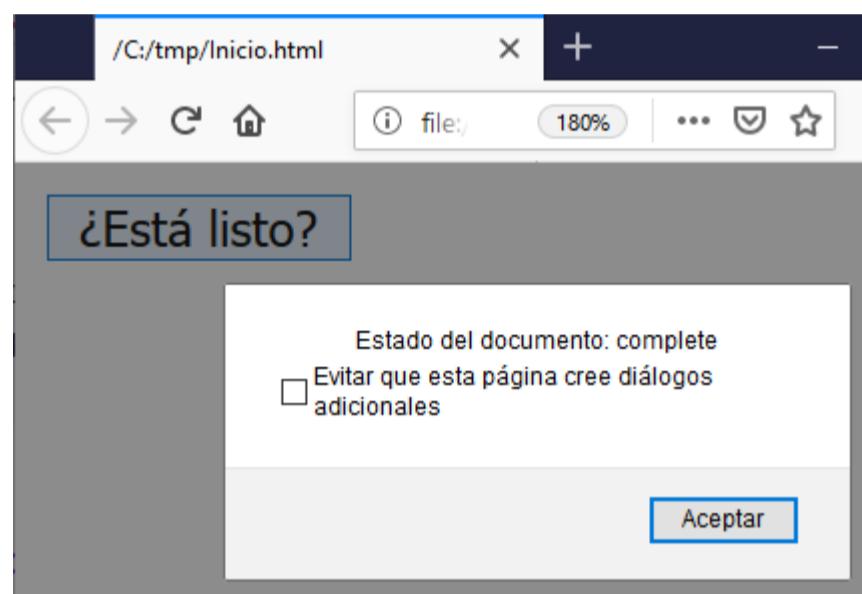


Ilustración 225: Muestra el estado del documento. Si ha cargado completamente es "complete"

```
<!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>
```

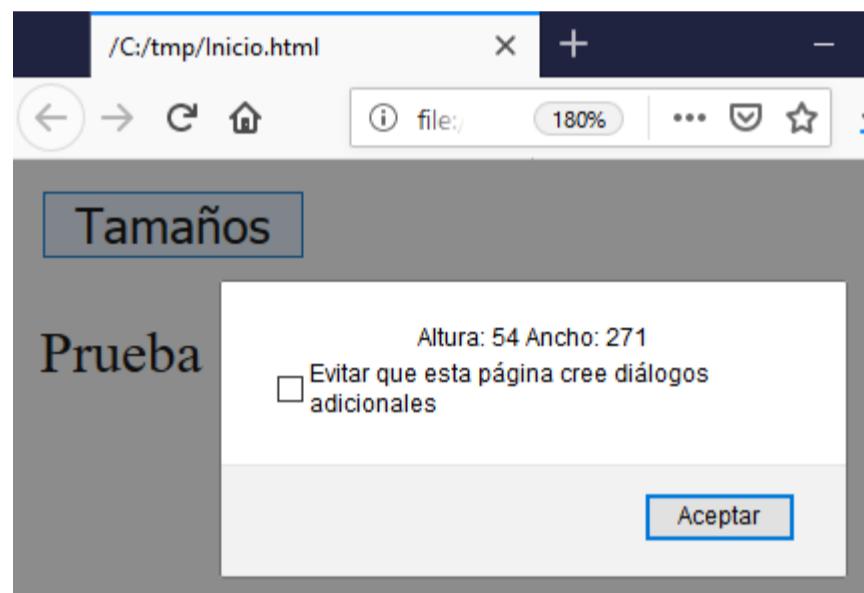


Ilustración 226: Alto y ancho de la página

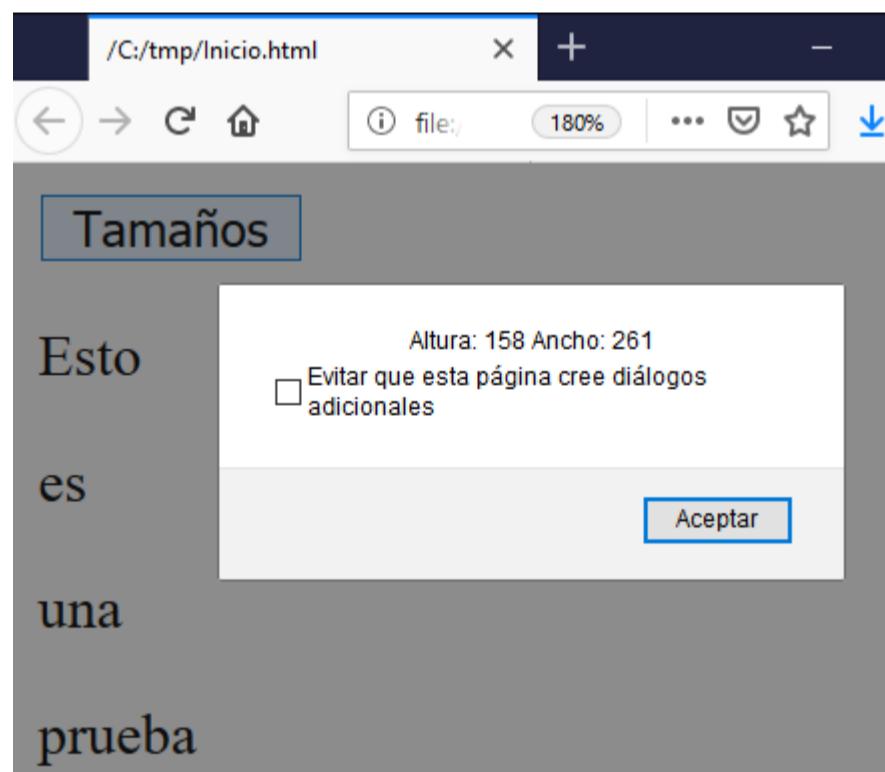
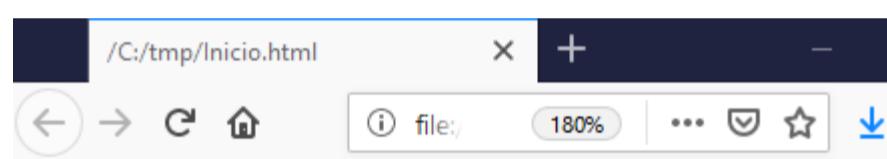


Ilustración 227: Alto y ancho de la página adicionando más <p>

```
<!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>
```



Mi página web
Libro de Redes Neuronales
¿Enlaces?

Ilustración 228: Una página con dos enlaces

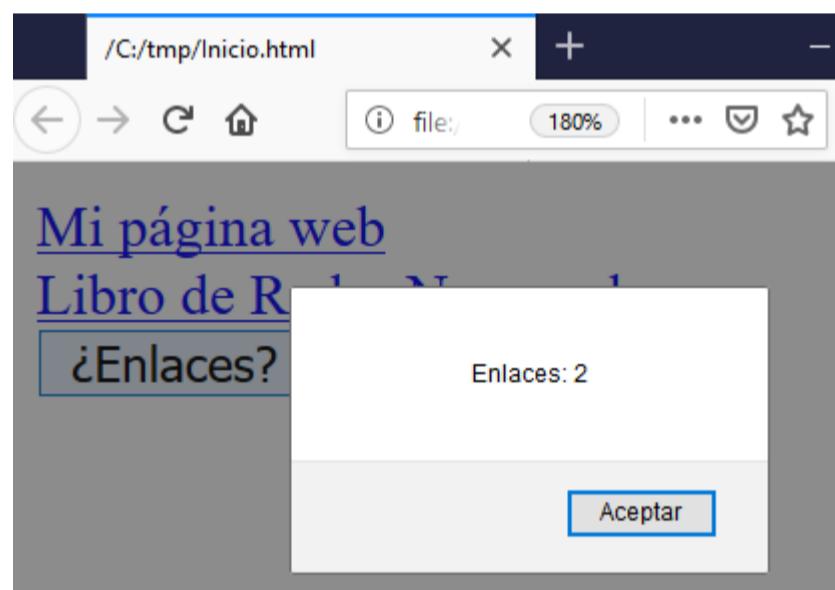


Ilustración 229: Cuenta el número de enlaces de la página

```
<!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/LibroRedNeuronal2020">Redes neuronales</a><br>
<a href="https://github.com/ramsoftware/Evaluador3">Evaluador de expresiones</a><br>
<input type="button" onclick="MuestraEnlaces()" value="¿Enlaces?">
</html>
```

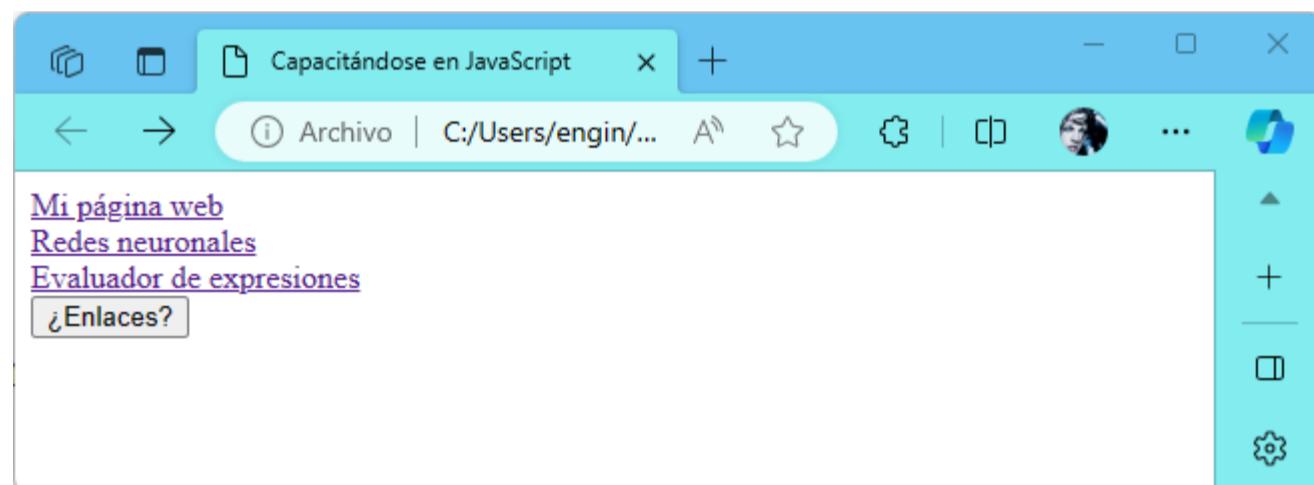


Ilustración 230: Una página con tres enlaces

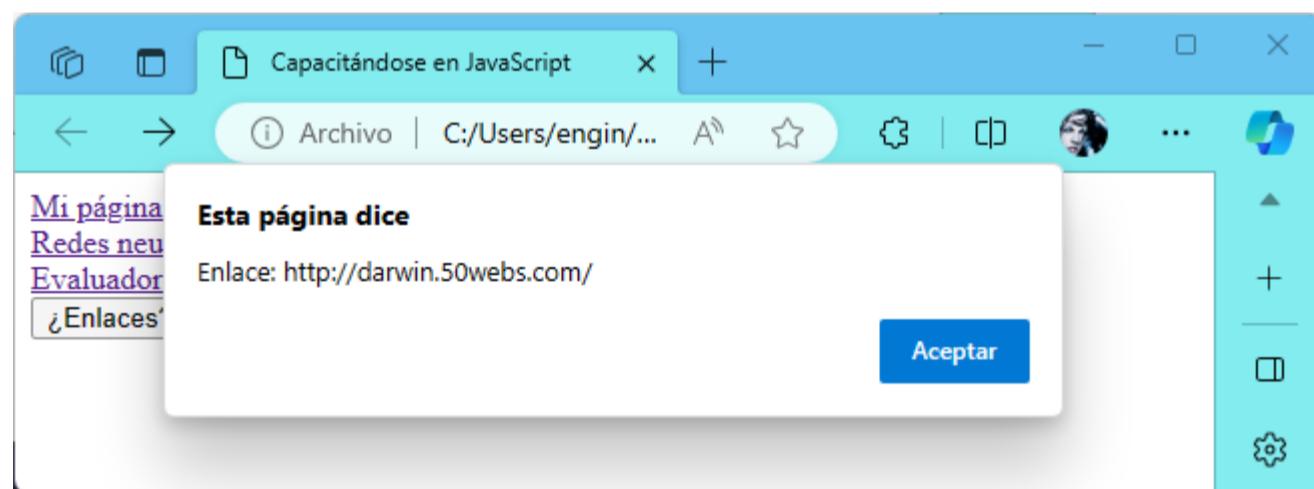


Ilustración 231: Muestra el enlace de cada página

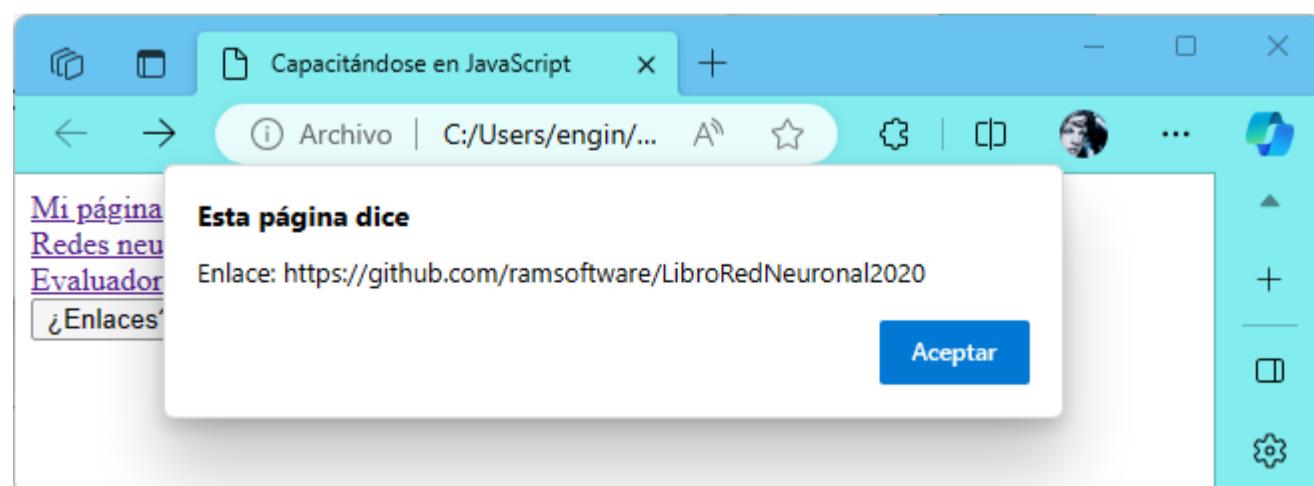


Ilustración 232: Muestra el enlace de cada página

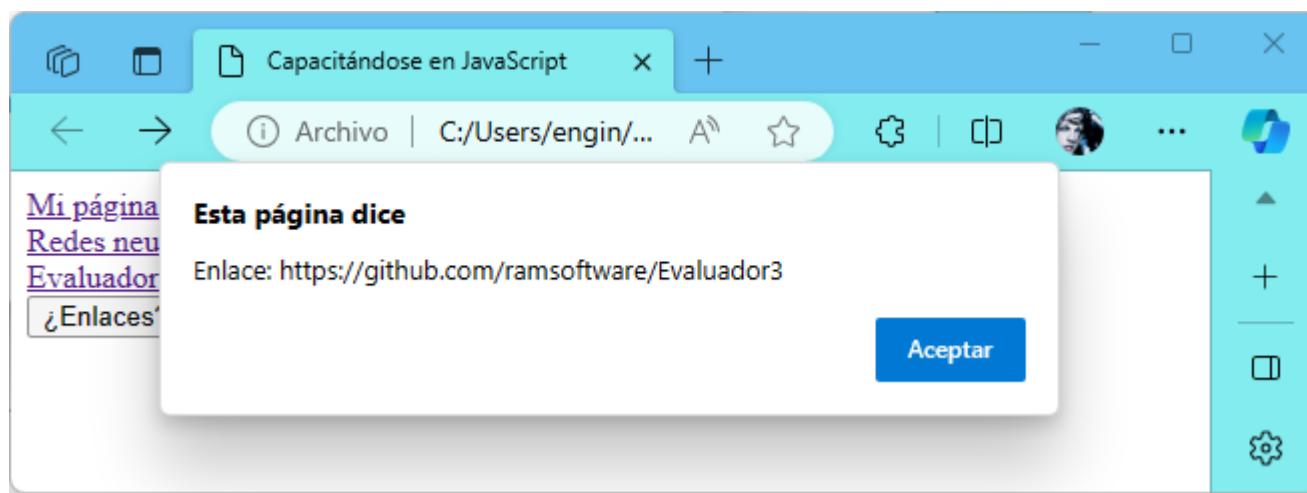


Ilustración 233: Muestra el enlace de cada página

```
<!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>
```

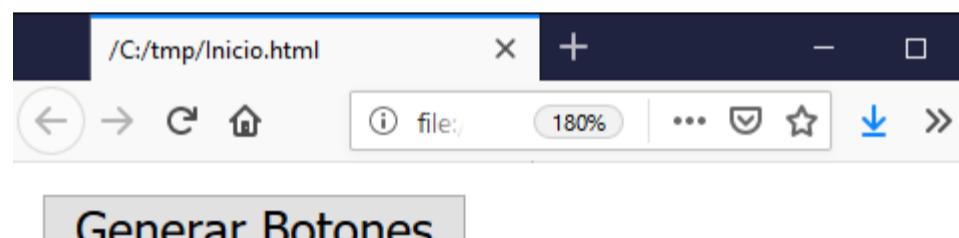


Ilustración 234: Un documento con un botón

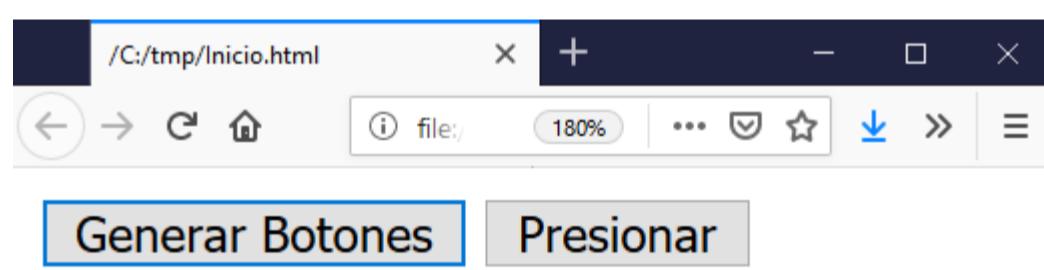


Ilustración 235: Al presionar el botón se genera un nuevo botón

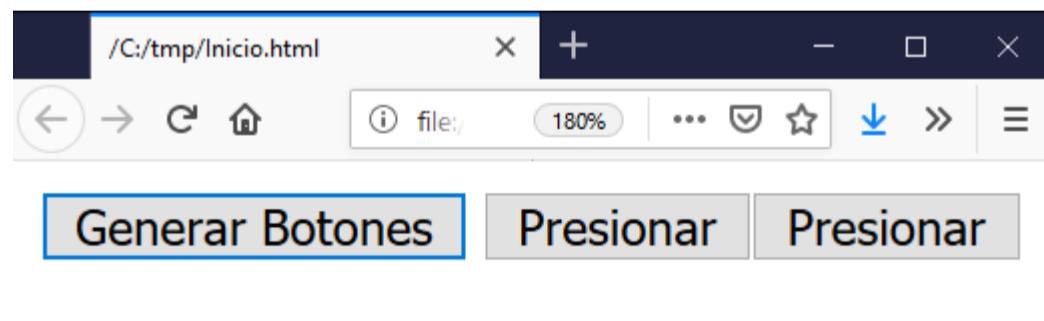


Ilustración 236: Y se pueden generar más botones en tiempo de ejecución

```

<!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>

```

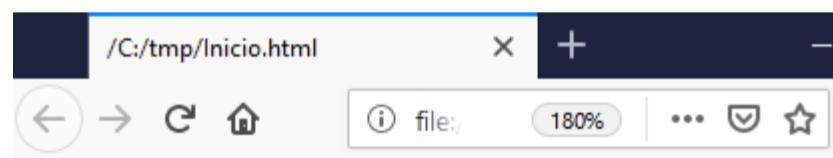
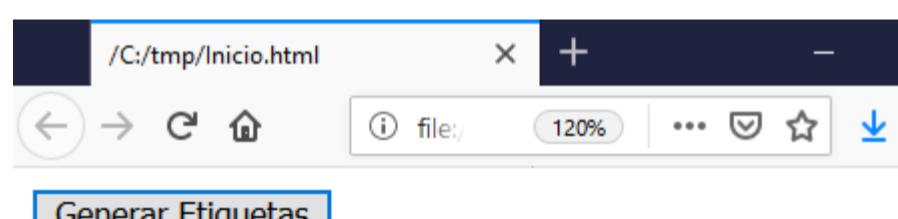
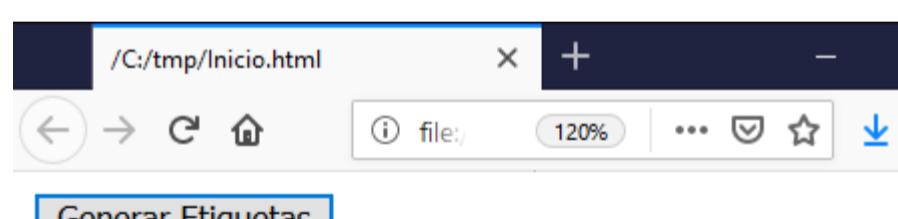


Ilustración 237: Un documento con un botón



Este es un texto

Ilustración 238: Al presionar el botón se genera un nuevo <p>



Este es un texto

Este es un texto

Ilustración 239: Se generan nuevos <p> en tiempo de ejecución

```

<!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>

```

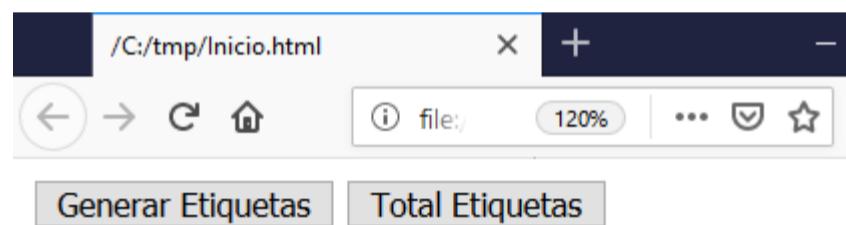
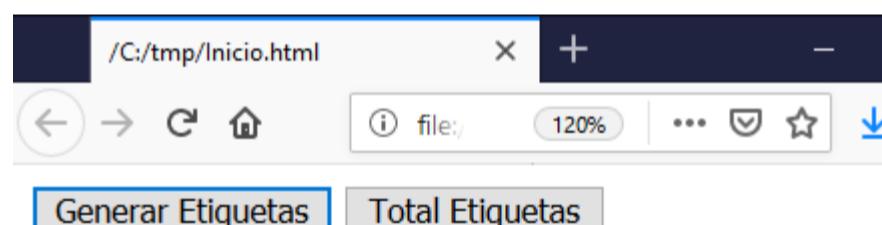


Ilustración 240: Un documento con dos botones



Este es un texto

Este es un texto

Ilustración 241: Se generan dos <p> en tiempo de ejecución

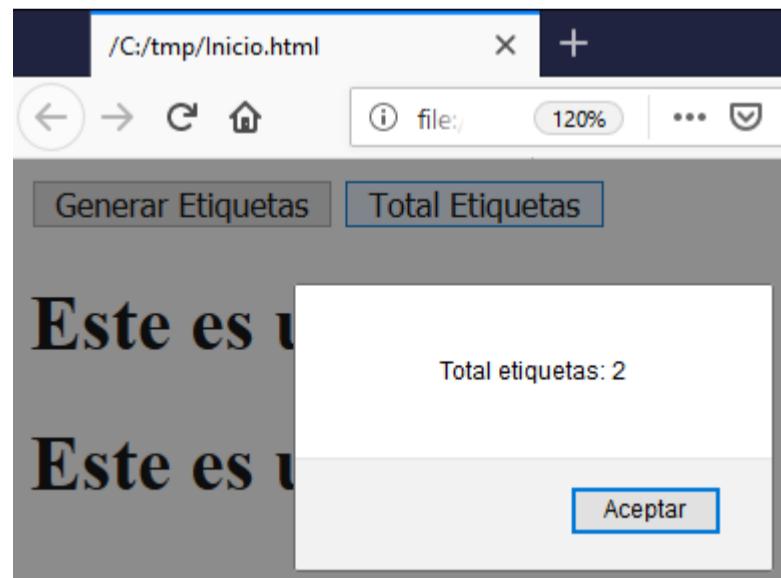


Ilustración 242: Se cuenta el número de <p> en el documento

```
<!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>
```

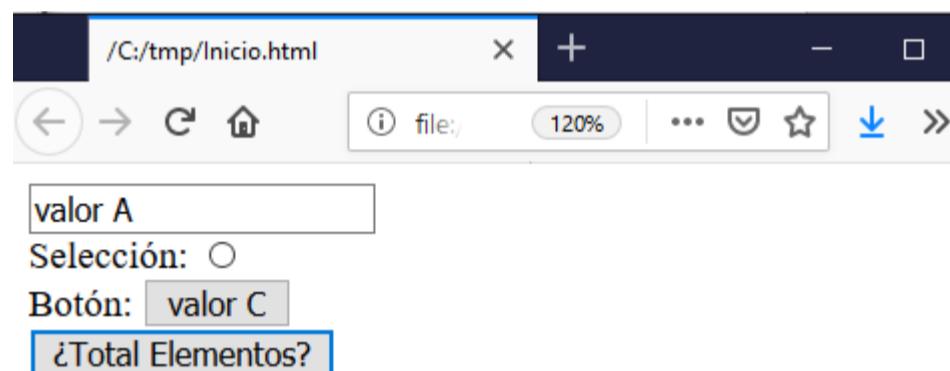


Ilustración 243: Varios tipos de objeto en la página

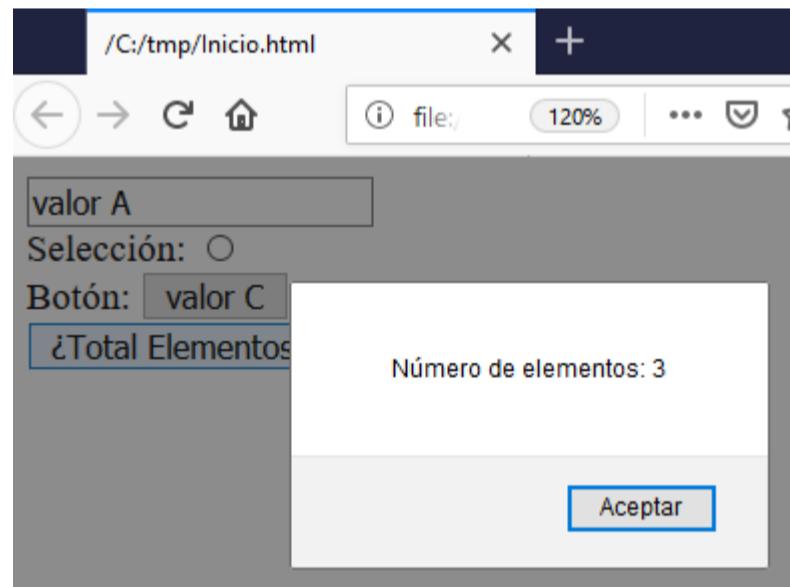


Ilustración 244: Cuenta el número de objetos en el documento

```
<!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>
```

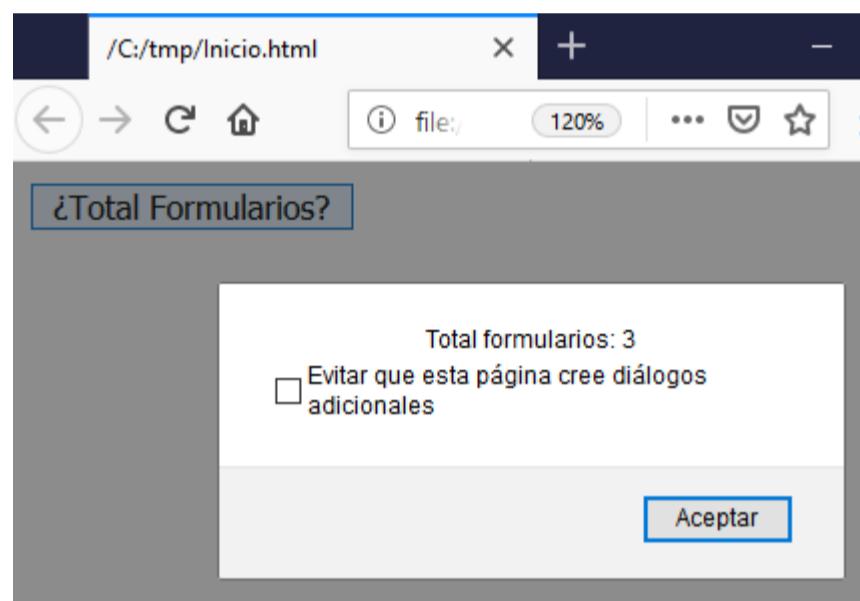


Ilustración 245: Cuenta el número de formularios en el documento

```
<!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>
```

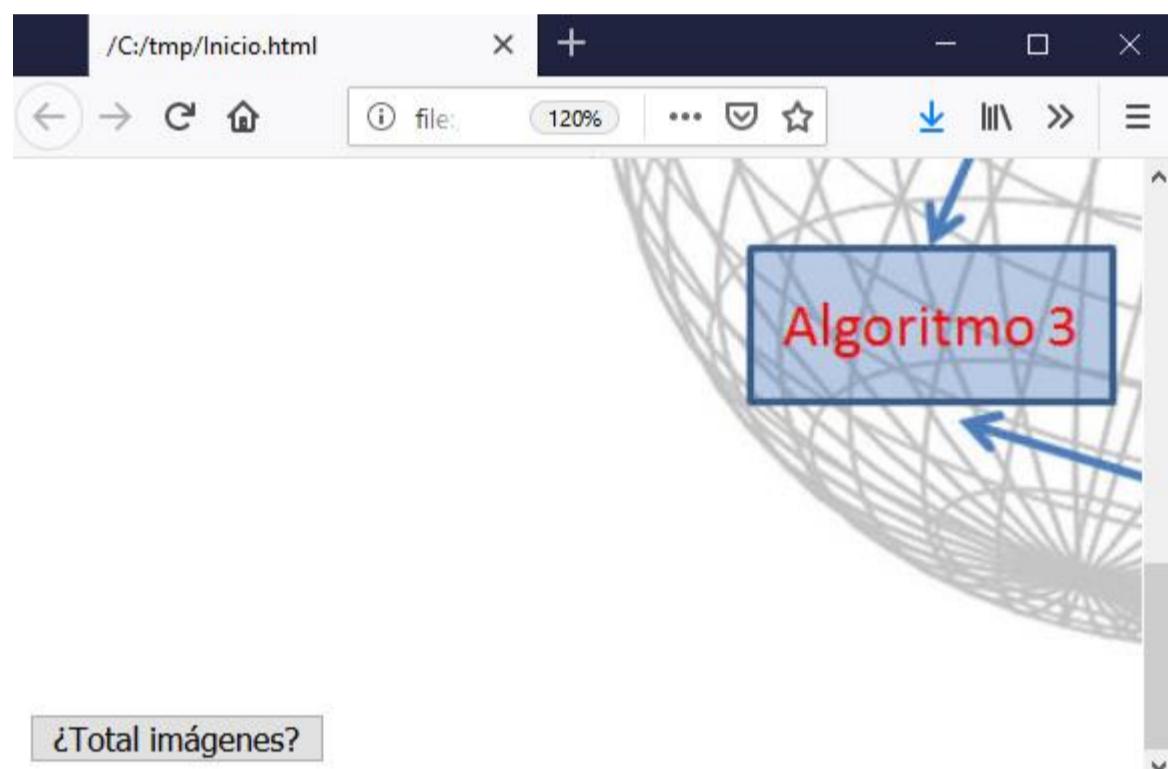


Ilustración 246: Un documento con una imagen

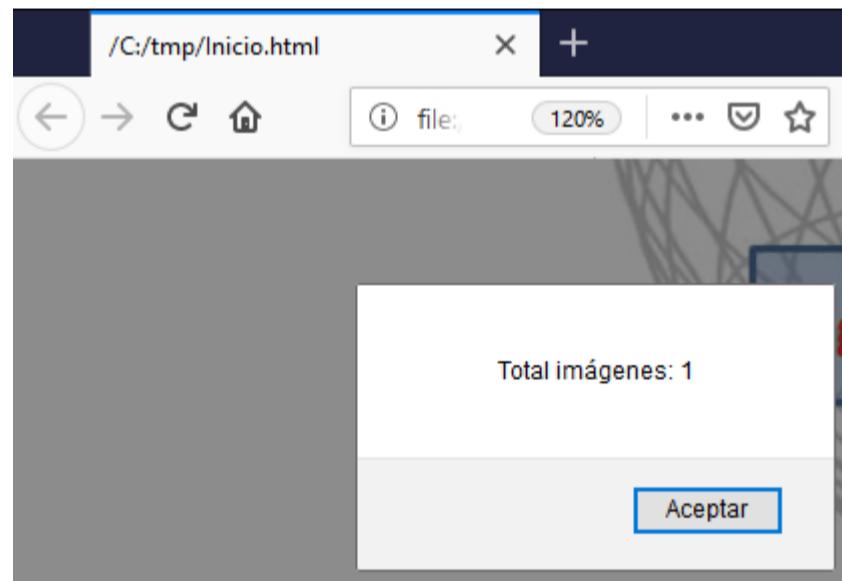


Ilustración 247: Cuenta el número de imágenes en el documento

```
<!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>
```

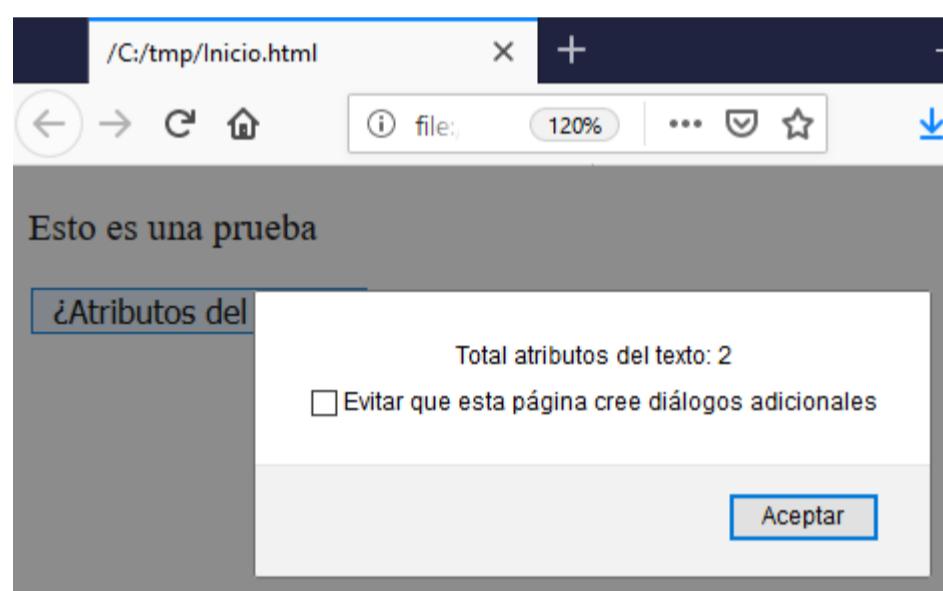


Ilustración 248: Cuenta el número de atributos que tenga un objeto

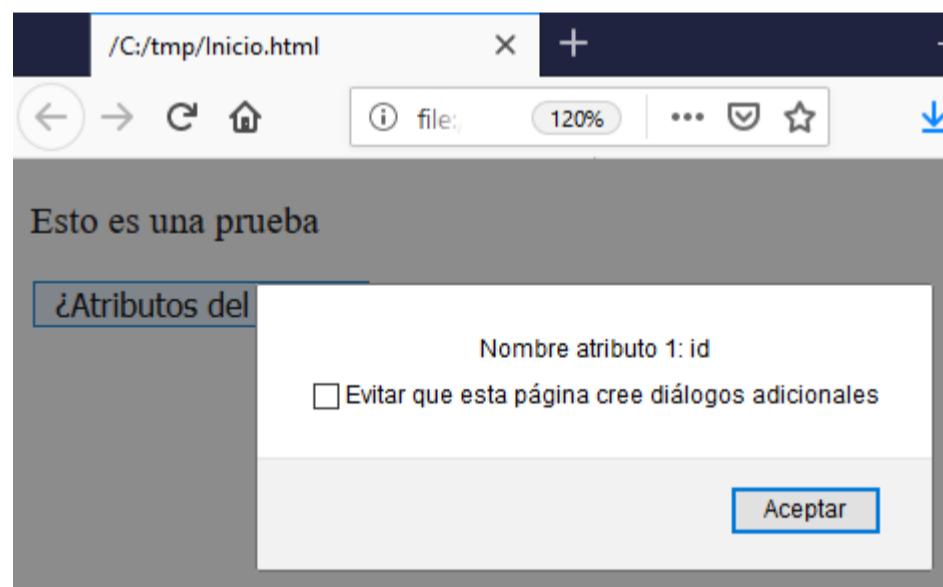


Ilustración 249: Trae la identificación de un objeto

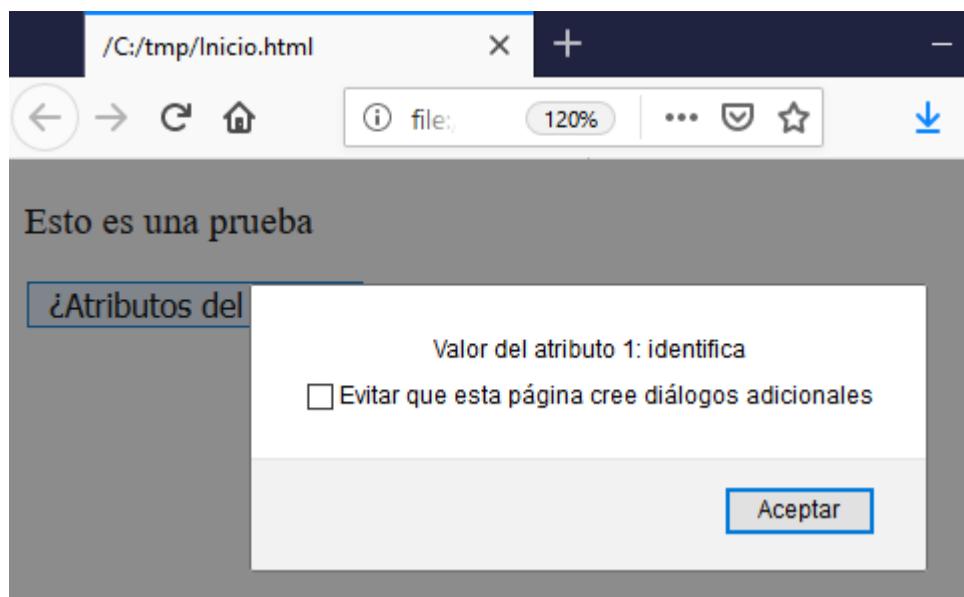


Ilustración 250: Trae el valor de la identificación de ese atributo

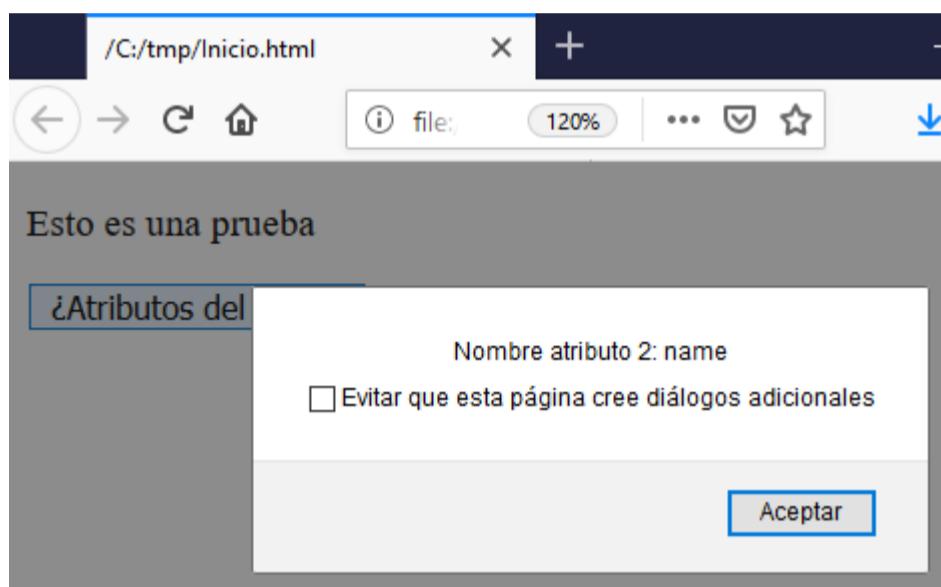


Ilustración 251: Trae otro atributo

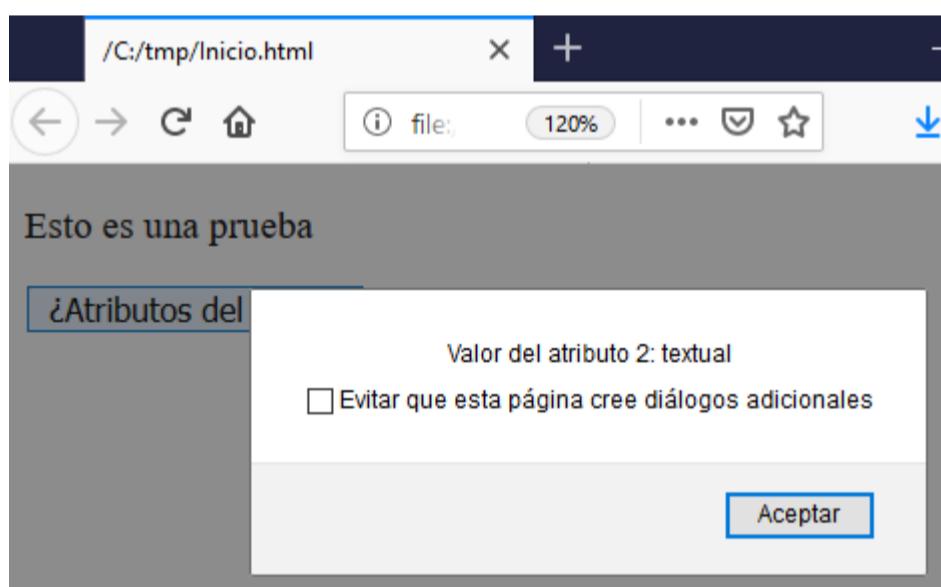


Ilustración 252: Muestra el valor de ese segundo atributo

```
<!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>
```



Ilustración 253: Abre una nueva ventana y apunta a una determinada dirección

```
<!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>
```

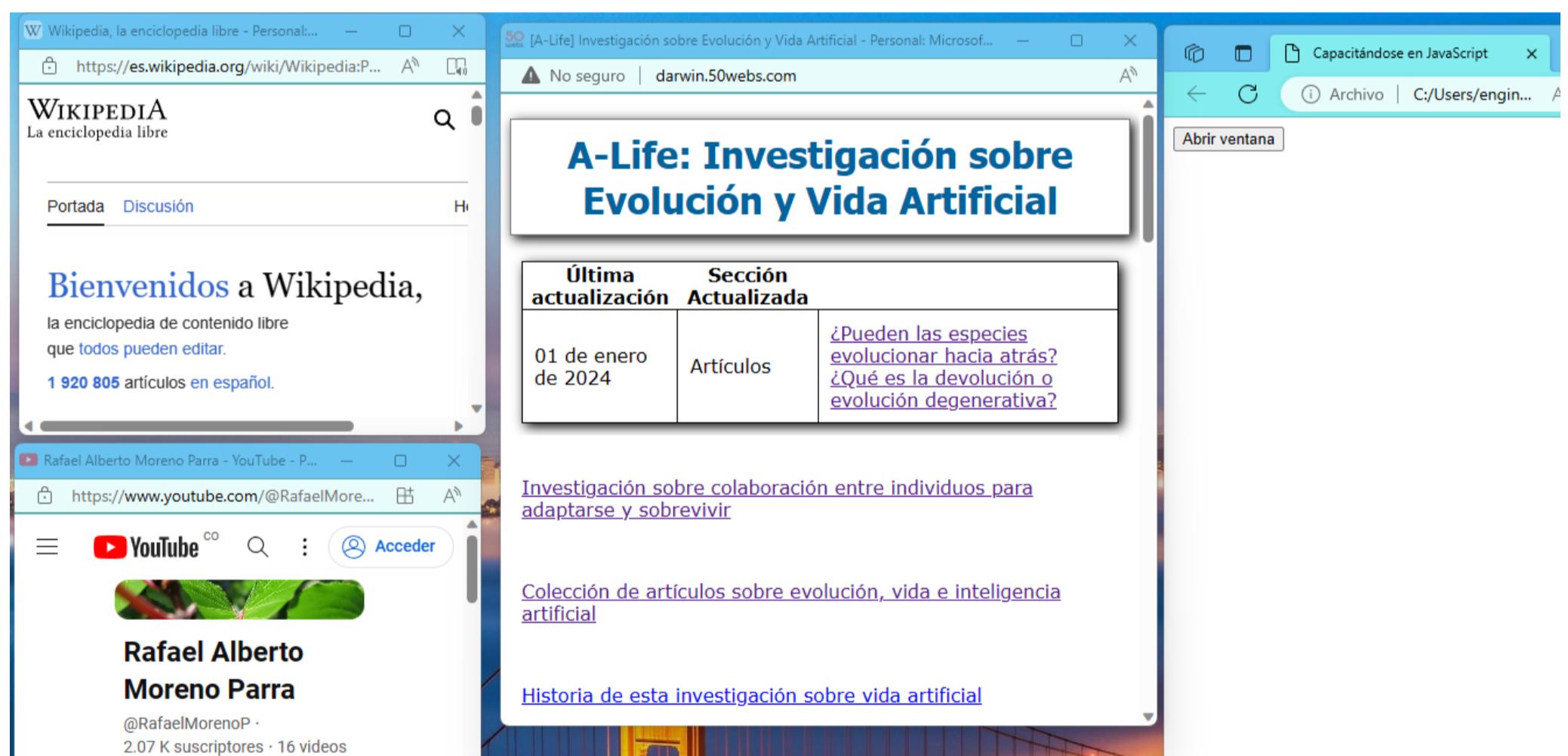


Ilustración 254: Abre varias ventanas, cada una hacia una dirección distinta

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>
```

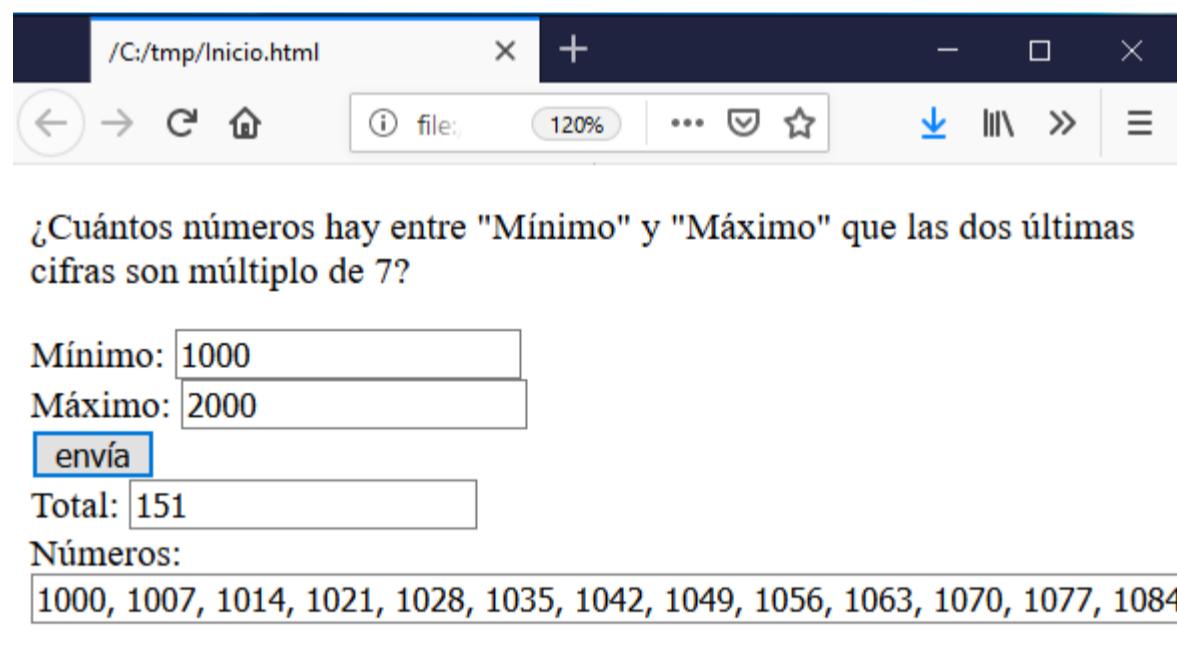


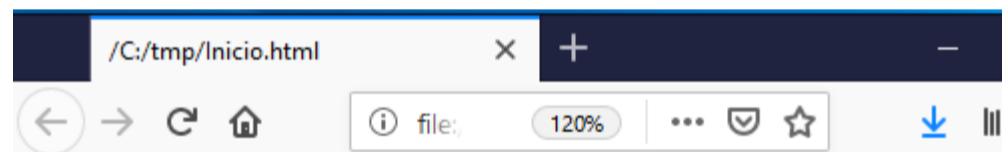
Ilustración 255: Un formulario que ejecuta un algoritmo local al enviar la información

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

```

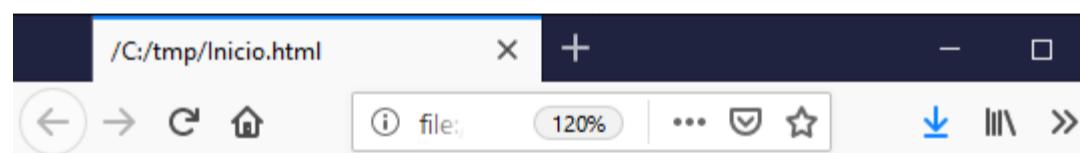


Ingrese dirección de correo electrónico

Correo:

 Resultado:

Ilustración 256: Valida el correo electrónico



Ingrese dirección de correo electrónico

Correo:

 Resultado:

Ilustración 257: Valida el correo electrónico

Valida una URL con expresiones regulares

255.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>
```

The screenshot shows a web browser window with the title bar reading "/C:/tmp/Inicio.html". The main content area displays the following form:

Ingrese una URL

URL: abcde

envía

Mensaje: Es errónea la URL

The "envía" button is highlighted in blue, indicating it was clicked.

Ilustración 258: Valida la URL

The screenshot shows a web browser window with the title bar reading "/C:/tmp/Inicio.html". The main content area displays the following form:

Ingrese una URL

URL: http://darwin.50webs.com

envía

Mensaje: La URL es correcta

The "envía" button is highlighted in blue, indicating it was clicked.

Ilustración 259: Valida la URL

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="miliendo" 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="milienzo" width="200" height="200" style="border:1px solid #000000;"></canvas>
<script>
    //Funciones para hacer gráficos
</script>
</body>
</html>
```

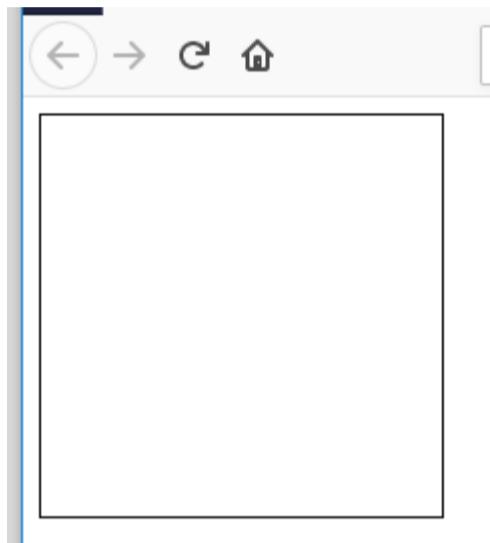


Ilustración 260:Lienzo ("canvas") visible usando un borde

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="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.stroke(); //Hace visible la línea
</script>
</body>
</html>
```



Ilustración 261: Línea dibujada

```
<!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.stroke(); //Hace visible la línea
</script>
</body>
</html>
```



Ilustración 262: 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 263: 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 264: 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>
```

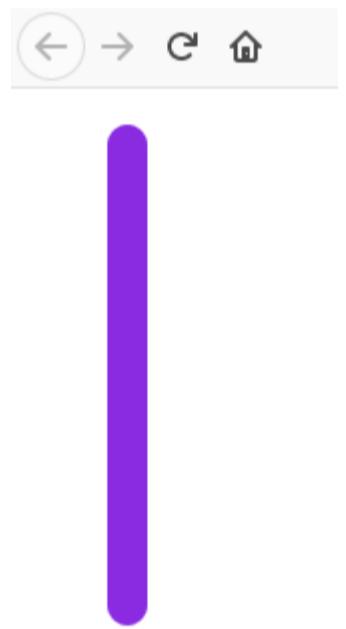


Ilustración 265: Las líneas finalizan redondeadas en ambos extremos usando "round"

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>
```

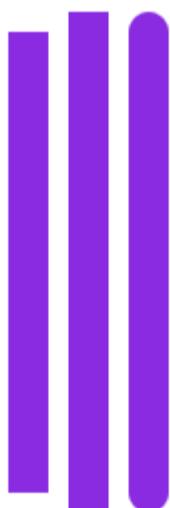


Ilustración 266:Tres líneas con la misma longitud, pero diferente cierre de extremo

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>
```

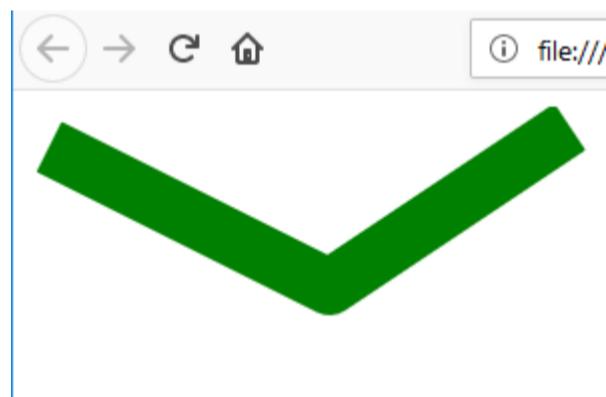


Ilustración 267: Dos líneas juntas con un empalme redondeado

```
<!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>
```

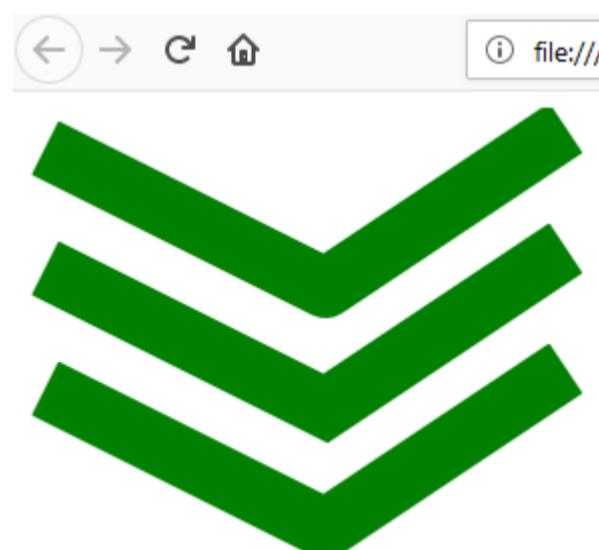


Ilustración 268: Diferentes formas de juntar dos líneas: "round", "miter", "bevel"

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 linea
    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 linea donde termina la línea 3 y empieza la línea 1
    contexto.closePath();

    contexto.stroke(); //Dibuja
</script>
</body>
</html>
```

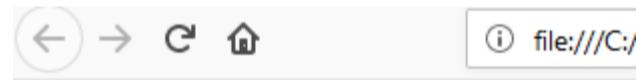


Ilustración 269: Una figura cerrada

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 270: 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>
```

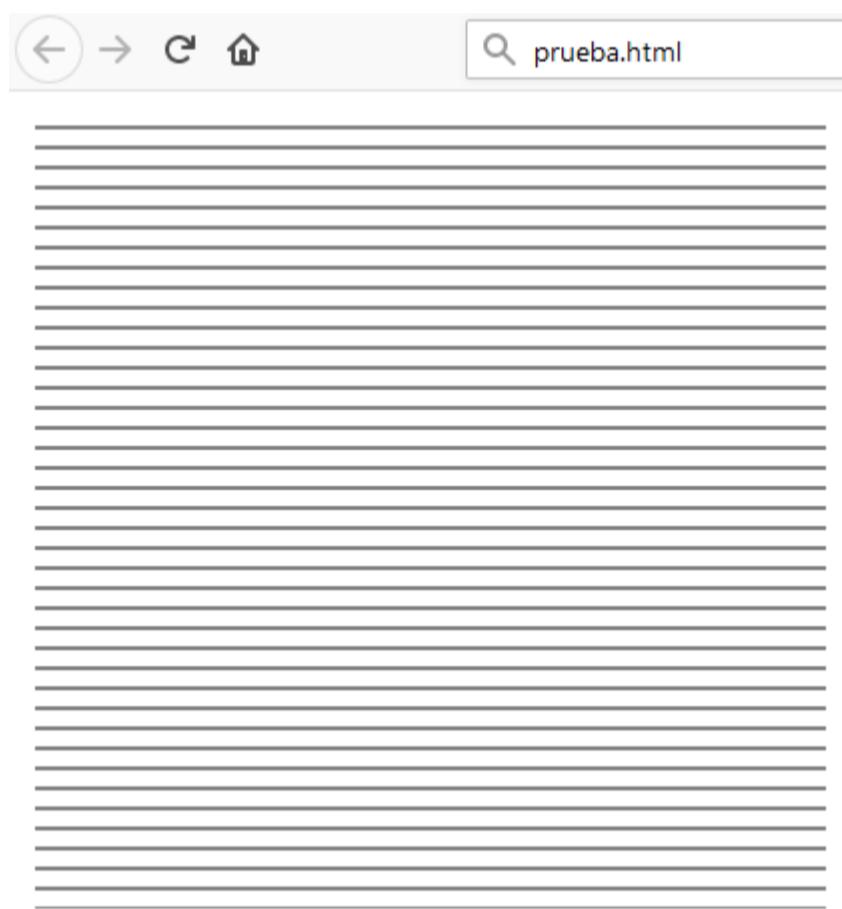


Ilustración 271: Líneas generadas por un ciclo

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>
```

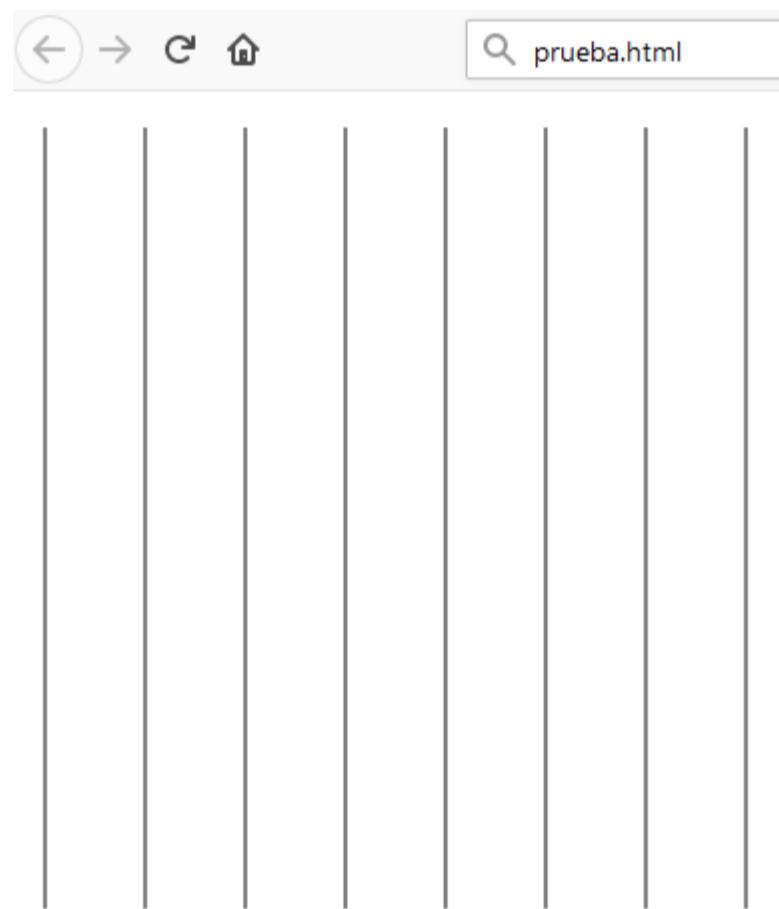


Ilustración 272: Líneas verticales dibujadas con un ciclo

```
<!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>
```

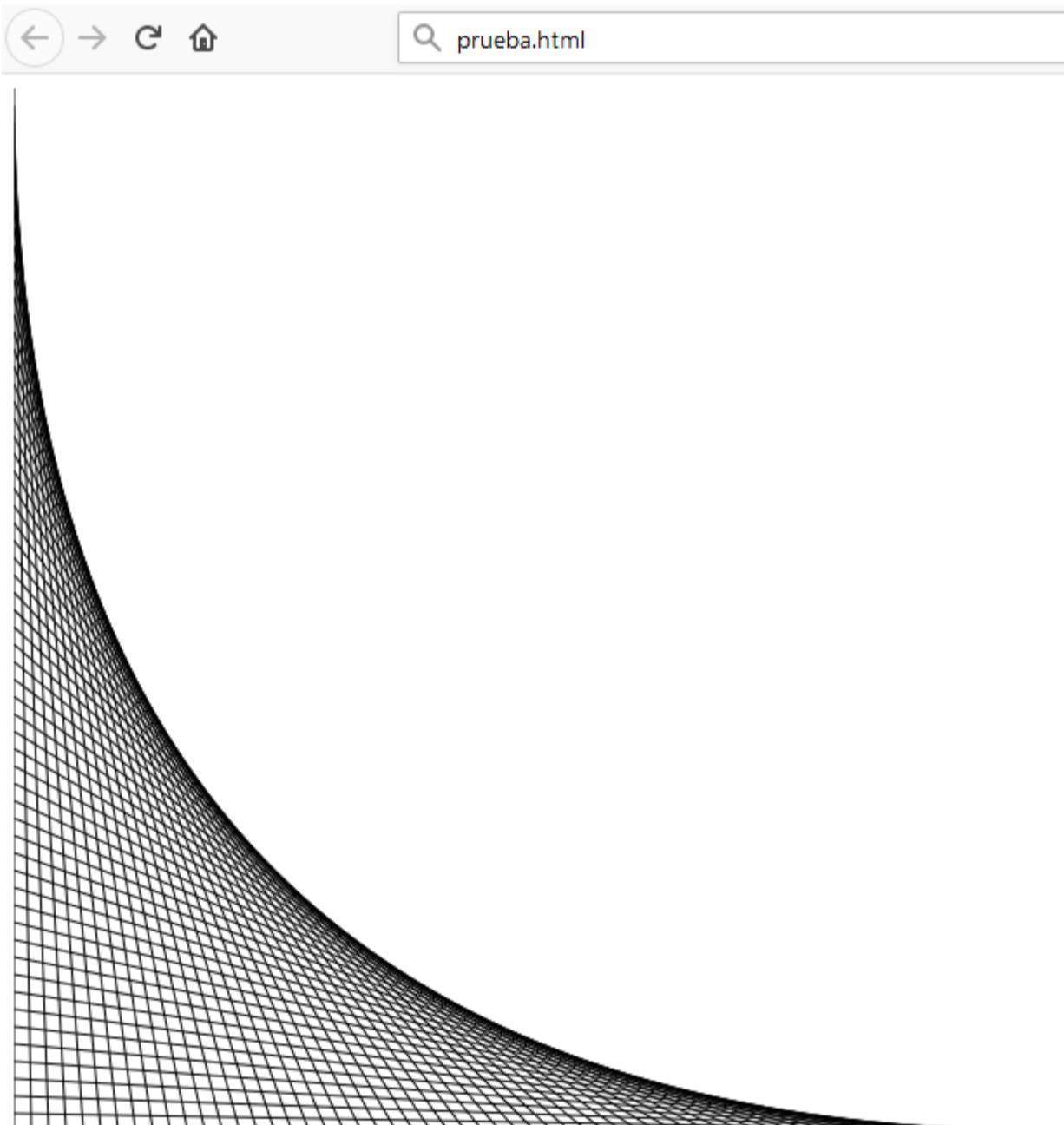


Ilustración 273: Ilusión de curva usando líneas rectas

“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>
```

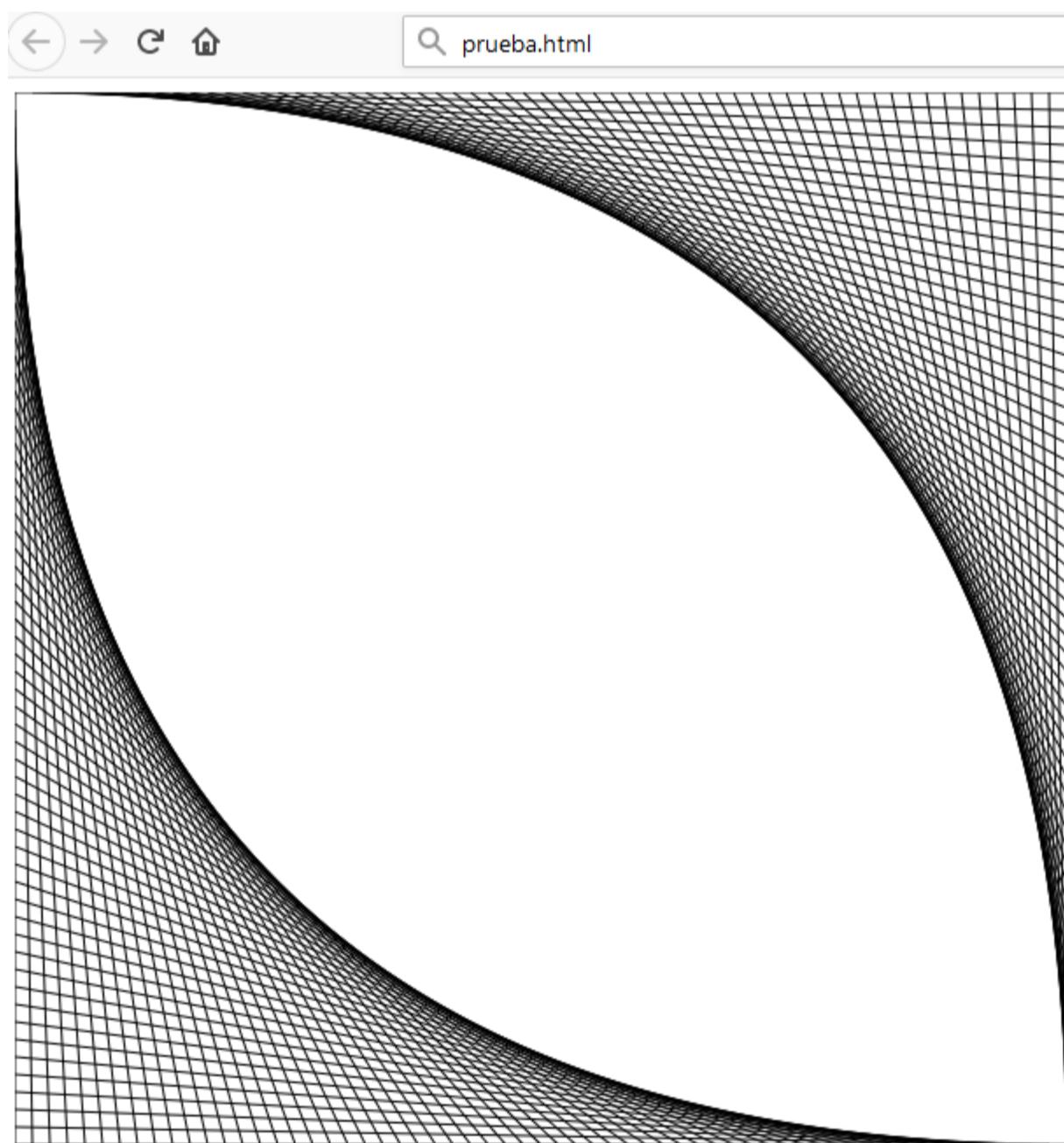


Ilustración 274: Ilusión de dos curvas usando líneas rectas

Dibujar círculos

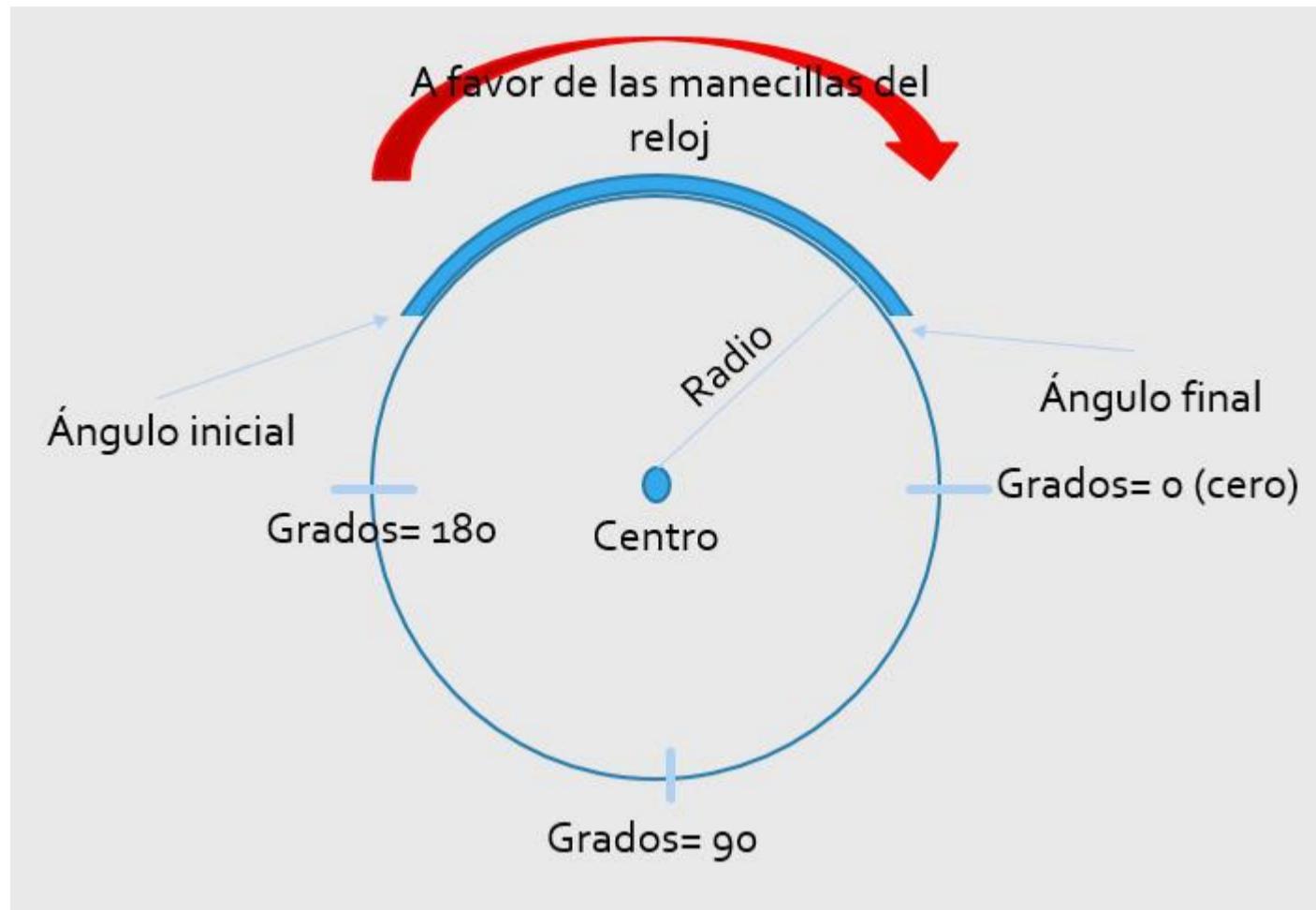


Ilustración 275: La manera en que se dibujará el círculo

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>
```

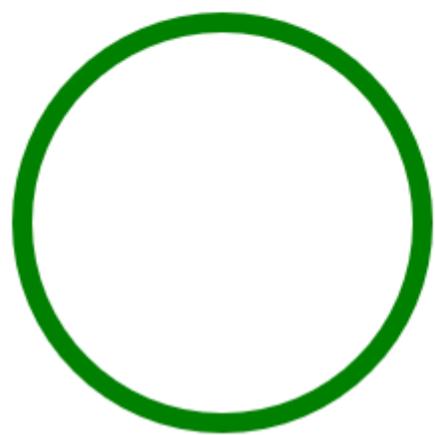


Ilustración 276: Círculo

Dibujar curvas

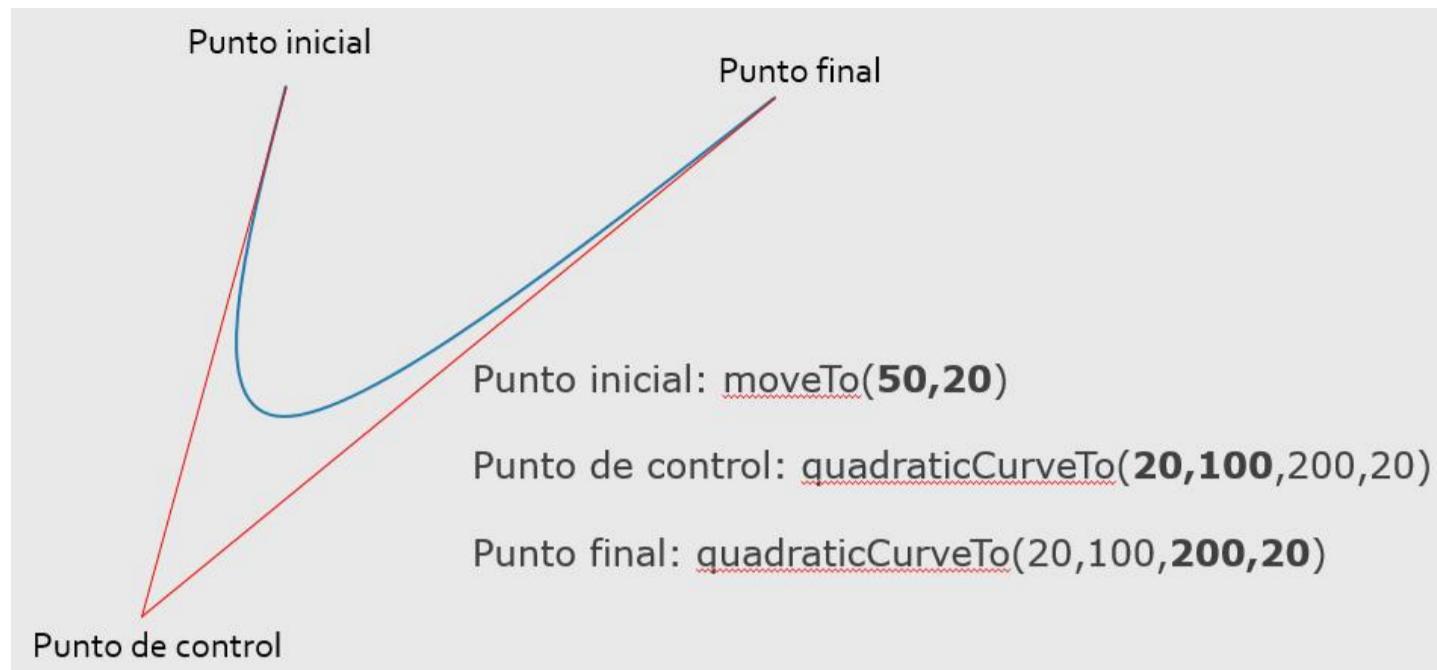


Ilustración 277: Forma en que se dibujarán las curvas

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>
```

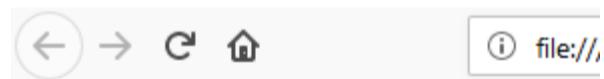
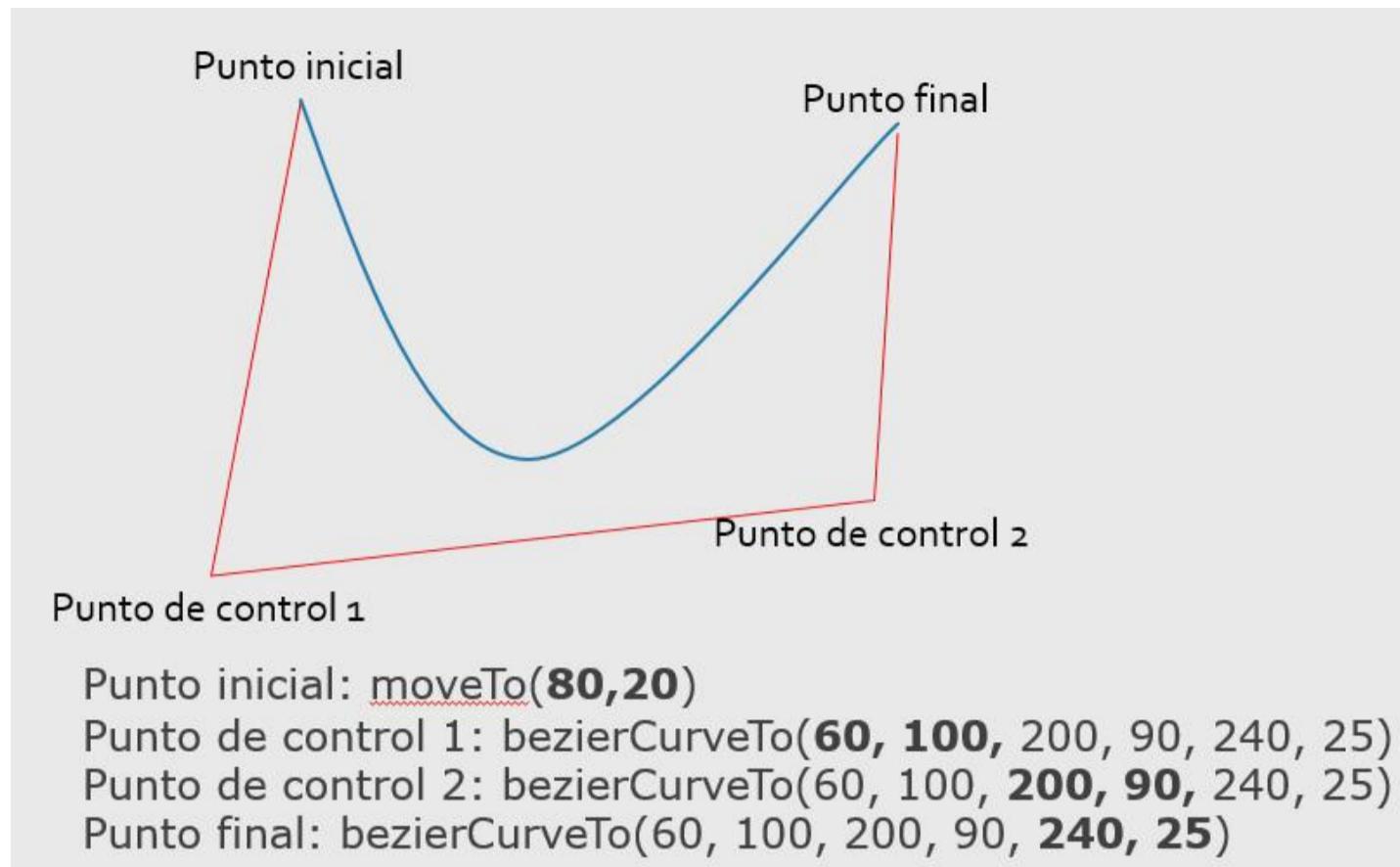


Ilustración 278: Curva

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>
```

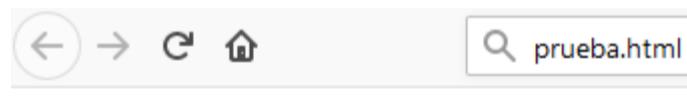


Ilustración 279: Curva Bézier

```
<!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>
```

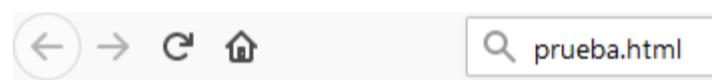


Ilustración 280: Combinando curvas

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>
```

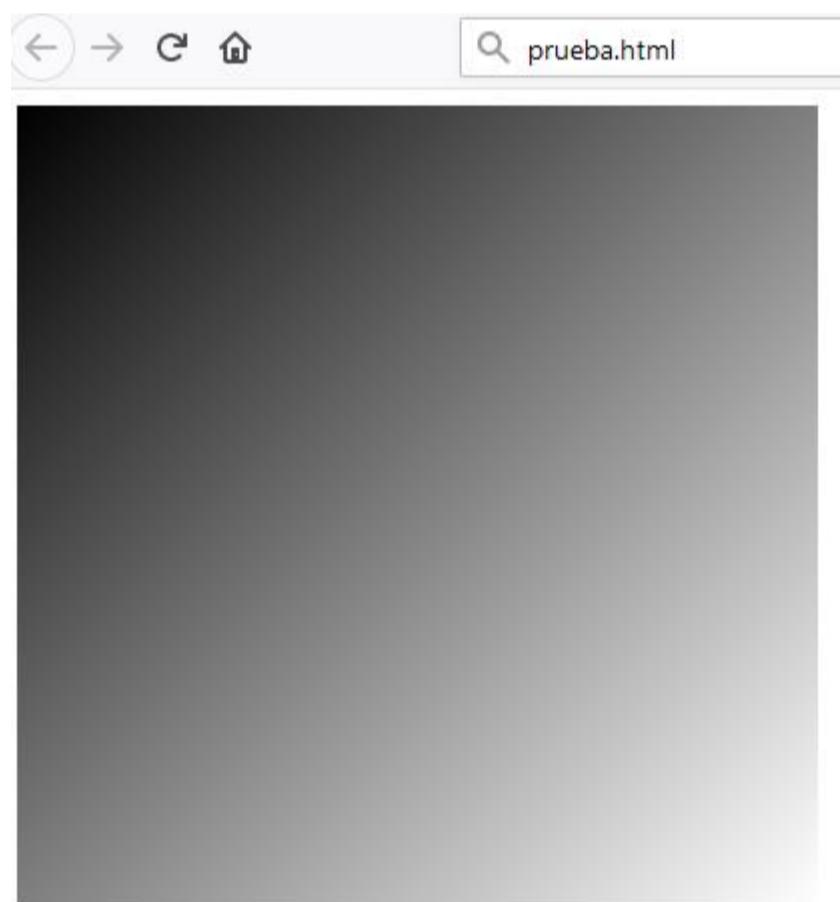


Ilustración 281: Variación del color

```
<!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>
```

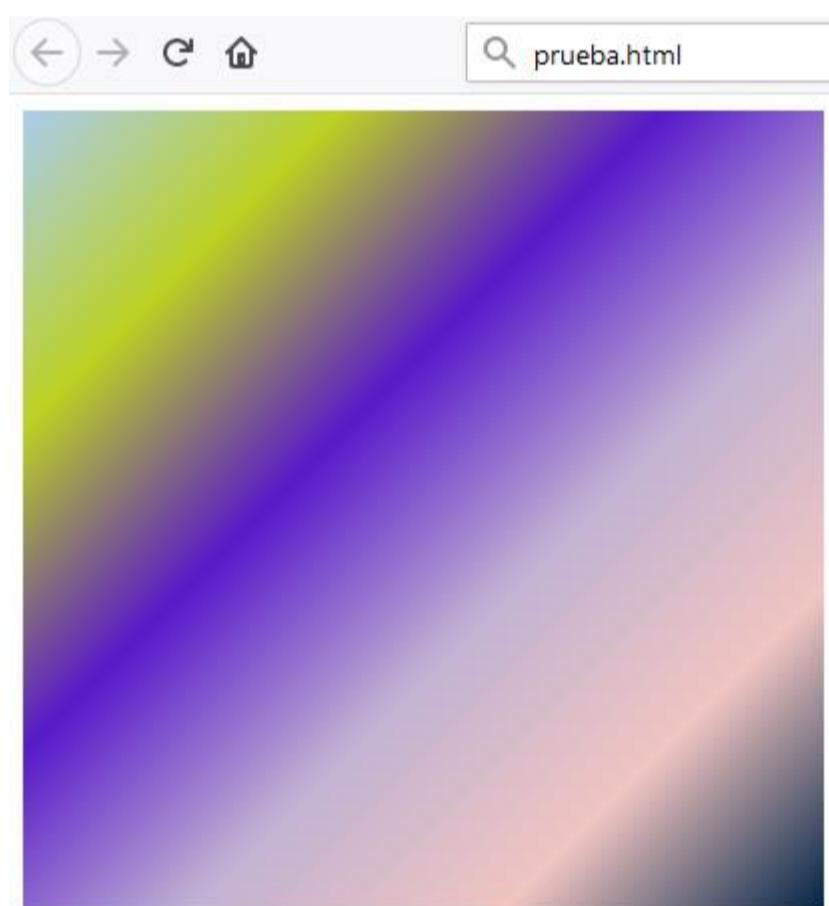


Ilustración 282: Variación de varios colores en forma oblicua

```
<!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>
```

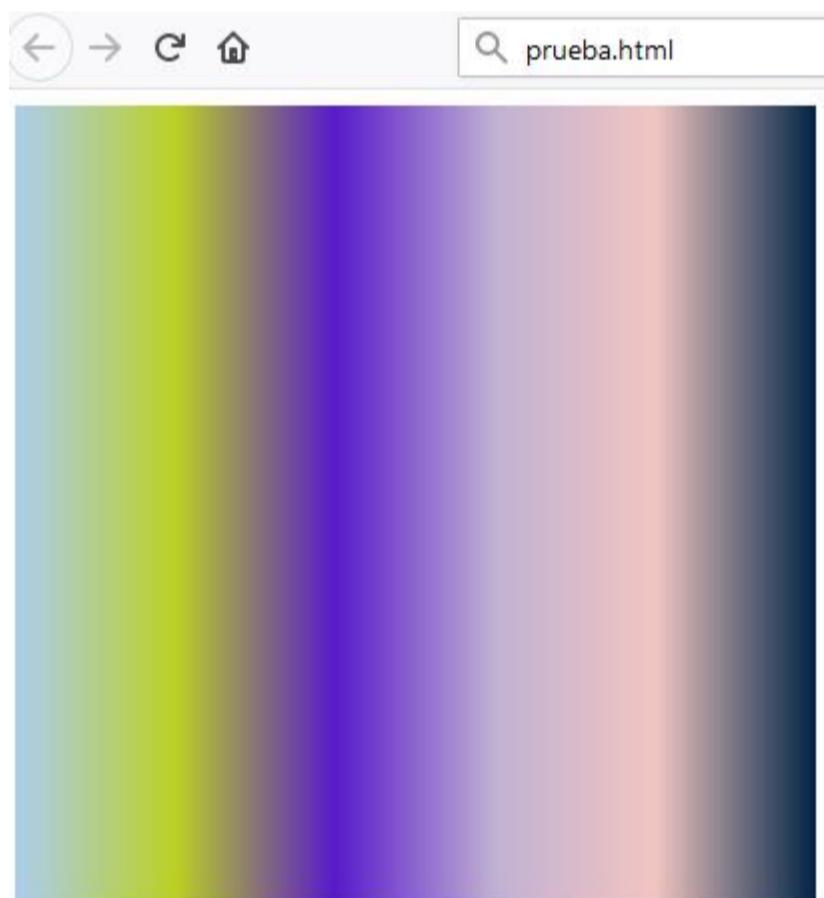


Ilustración 283: Variación de color en forma vertical

```
<!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>
```

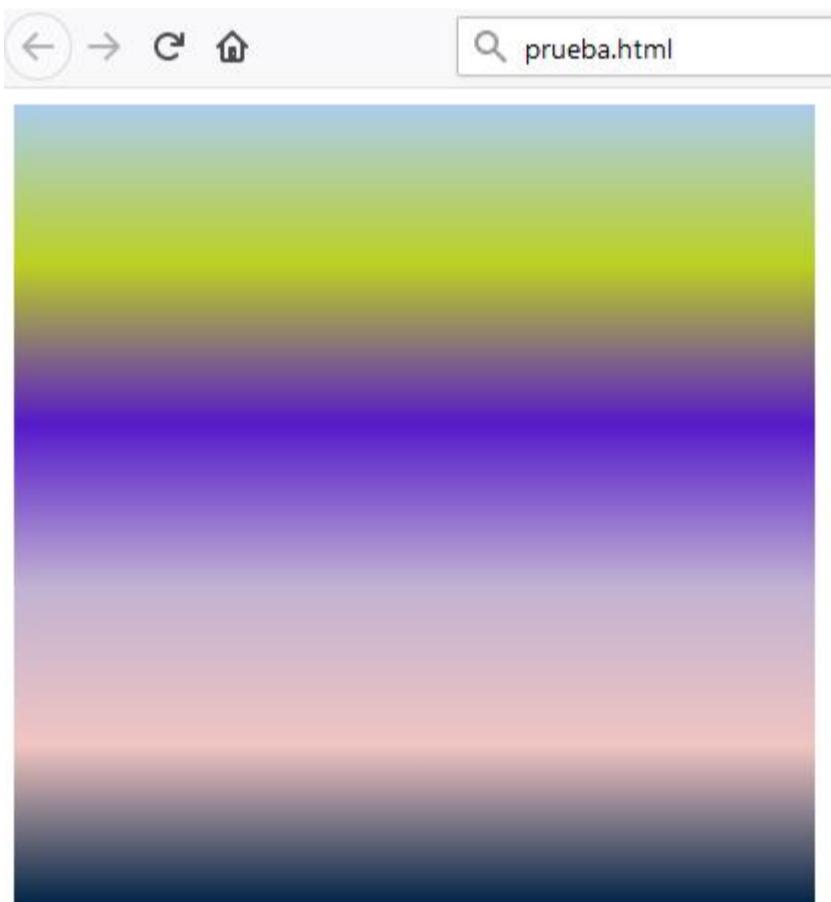


Ilustración 284: Variación de color horizontal

```
<!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>
```

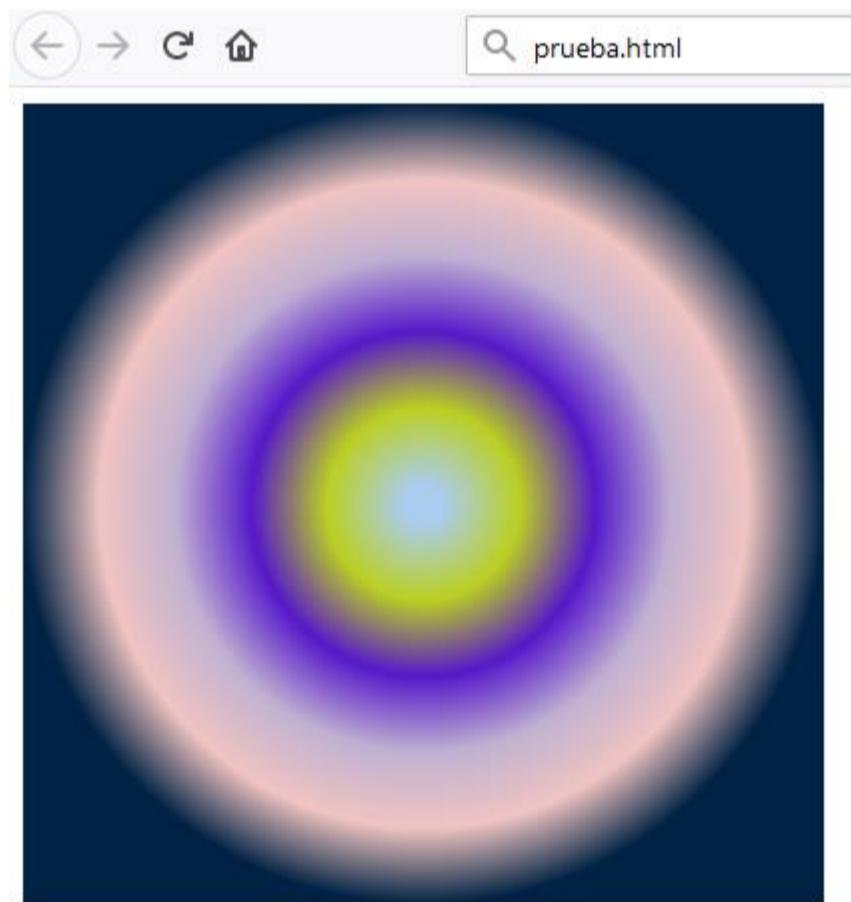


Ilustración 285: Variación en círculos

```
<!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 286: 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 287: Imagen original

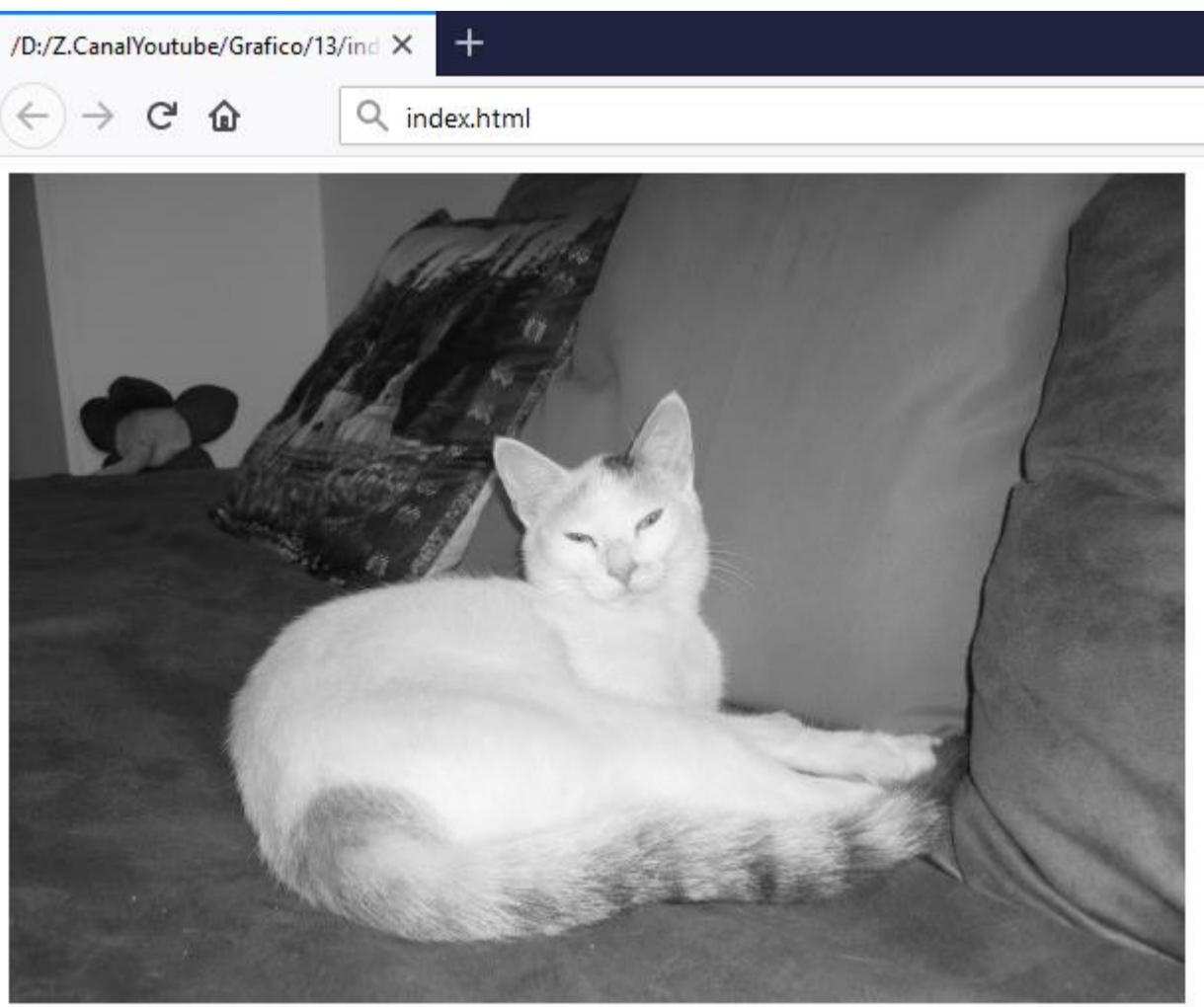


Ilustración 288: Imagen en escala de grises

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

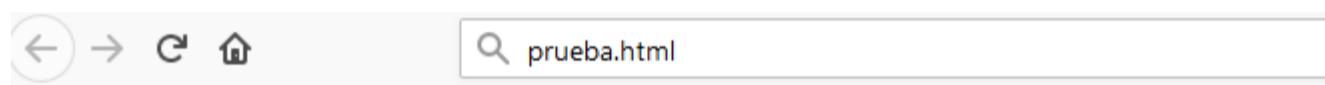
Ilustración 289: Dibuja un texto

```
<!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 290: Dibuja un texto con relleno de color

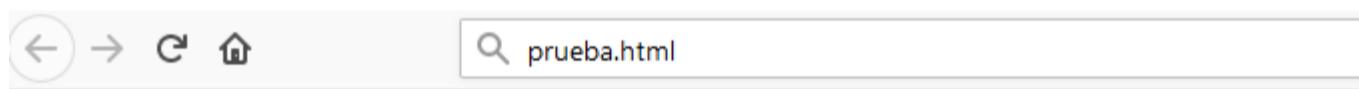
```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Capacitándose en JavaScript</title>
</head>
<body>
<canvas id="milienzo" width="700" height="400"></canvas>
<script>
    let lienzo = document.getElementById('milienzo');
    let contexto = lienzo.getContext('2d');

    //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 291: 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>
```

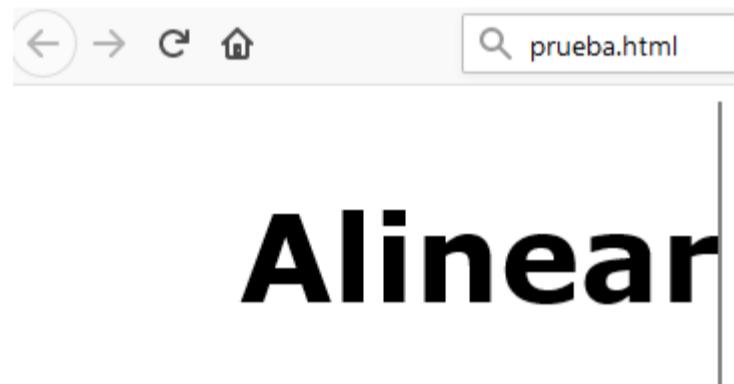


Ilustración 292: Dibuja un texto alineado al final

```
<!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 293: 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>
```

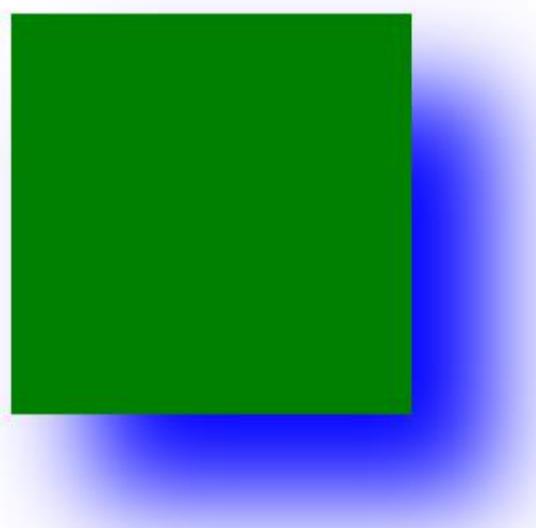


Ilustración 294: Sombra

```
<!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>
```

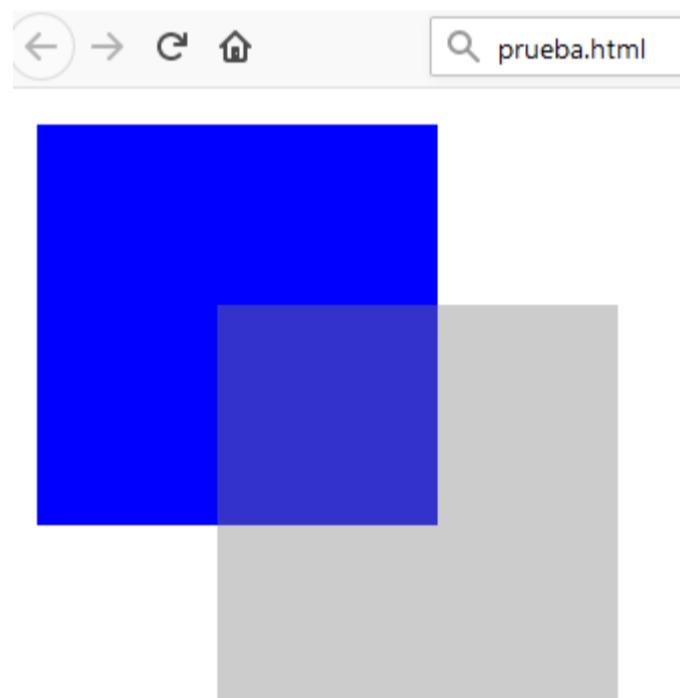


Ilustración 295: Rectángulo semitransparente

Referencias

- [1] W3Schools, «JavaScript Tutorial,» enero 2024. [En línea]. Available: <https://www.w3schools.com/js/>. [Último acceso: 2 enero 2024].
- [2] Mozilla, «JavaScript,» enero 2024. [En línea]. Available: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. [Último acceso: 2 enero 2024].
- [3] W3Techs, «Usage of client-side programming languages for websites,» enero 2024. [En línea]. Available: https://w3techs.com/technologies/overview/client_side_language/all. [Último acceso: 02 enero 2024].
- [4] Adobe, «Flash,» enero 2024. [En línea]. Available: <https://www.adobe.com/products/flashplayer/end-of-life-alternative.html>. [Último acceso: 02 enero 2024].
- [5] ECMA International, «ECMAScript® 2023 language specification,» junio 023. [En línea]. Available: <https://ecma-international.org/publications-and-standards/standards/ecma-262/>. [Último acceso: 2 enero 2024].
- [6] W3Schools, «JavaScript versions,» 2024. [En línea]. Available: https://www.w3schools.com/js/js_2023.asp. [Último acceso: 02 enero 2024].