

STRING

fromCharCode(num) : es un método estático que se utiliza para crear una cadena de texto a partir de uno o más valores numéricos que representan los códigos de caracteres en el conjunto de caracteres Unicode (o ASCII en su mayoría).

Ej:

```
let letra = String.fromCharCode(65); // Código Unicode para 'A'  
console.log(letra); // Salida: "A"
```

substring(start, end) : se utiliza para extraer una porción de una cadena de texto entre dos índices especificados, sin modificar la cadena original.

Ej:

```
let texto = "JavaScript";  
let subtexto = texto.substring(0, 4); // Extrae los caracteres desde el índice 0 hasta el índice 4 (sin incluirlo)  
console.log(subtexto); // Salida: "Java"
```

charCodeAt(pos) : se utiliza para obtener el **código de carácter Unicode** (un número entero) de un carácter en una posición específica de una cadena de texto.

Ej:

```
let texto = "JavaScript";  
let codigo = texto.charCodeAt(0); // Obtiene el código del primer carácter, que es 'J'  
console.log(codigo); // Salida: 74
```

NUMBER

abs(): valor absoluto

random(): devuelve de 0 hasta 1 exclusivo

floor(): redondea hacia abajo

round(): se redondea hacia arriba o hacia abajo si es mayor o menor de 0.5

max(): valor maximo en un conjunto de numeros

sqrt(): raiz cuadrada

min(): valor minimo en un conjunto de numeros

pow(): elevar numero a potencia (base, exponente)

WINDOW

alert(): es un método utilizado para mostrar un **mensaje de alerta** en una ventana emergente (popup) del navegador.

confirm(): se utiliza para mostrar un cuadro de diálogo de confirmación con un mensaje y dos botones: **Aceptar** y **Cancelar**.

find(): es un método de la **ventana del navegador** que se utiliza para **buscar** una cadena de texto dentro de la página web.

prompt(): se utiliza para mostrar un cuadro de diálogo en el navegador que permite al usuario ingresar un valor.

print(): se utiliza para abrir el cuadro de diálogo de impresión del navegador, permitiendo que el usuario imprima el contenido de la página web actual.

Interacción de los objetos con el navegador

- El objeto **Window** — Propiedades básicas:
 - Nombre de la ventana: Por defecto no tiene pero se le puede asignar
 - Tamaño de la ventana con toolbar y scrollbar:
 - Tamaño del viewport (sin toolbar y scrollbar):
 - Scroll horizontal y Vertical
 - Distancia desde la esquina superior izquierda:

```
window.name = "Mi Ventana";
```

```
window.outerHeight;  
window.outerWidth;
```

```
window.innerHeight;  
window.innerWidth;
```

```
window.pageXOffset; window.scrollX;  
window.pageYOffset; window.scrollY;
```

```
window.screenX;  
window.screenY;
```

- El objeto **Window** — Propiedades con iframes:
 - **window.frame**: Devuelve todos los elementos iframe de la ventana
 - **window.frameElement**: Devuelve el iframe en el que la ventana está insertada. Si no está insertada en un iframe devuelve null.
 - **window.length**: Devuelve el número de frames de la ventana
- El objeto **Window** — Propiedades con otras ventanas:
 - **nuevaVentana.closed**: Devuelve true si la ventana está cerrada
 - **nuevaVentana.opener**: Devuelve referencia con la ventana que creo la ventana actual.
 - **nuevaVentana.parent**: Devuelve la ventana padre de la ventana activa
 - **nuevaVentana.self**: Referencia a la ventana actual
- El objeto **Window** — Métodos:
 - **window.open()**: Abre una nueva ventana.

```
window.open(URL, name, specs, replace)
```
 - **window.close()**: Cierra una ventana.

```
window.close()
```
 - **window.resizeBy()**: Redimensiona una ventana un número de píxeles respecto a su tamaño actual. También esta `resizeTo` con valores totales.

```
resizeBy(width, height)
```
 - **windows.moveBy()**: Mueve una ventana una determinada cantidad de píxeles. **moveTo()** desplaza la ventana a una posición concreta.

```
window.moveBy(x, y)
```

Muchos más: **scrollBy()**....

- El objeto **Window** – Métodos - Instrucciones de tiempo:
El objeto window permite ejecutar código en intervalos de tiempo:

- **window.setTimeout()**: Ejecuta función pasados una determinada cantidad de ms.

`setTimeout(function, milliseconds, param1, param2, ...)`

- **window.clearTimeout()**: Cancela ejecución de un timeOut (retardo).

`clearTimeout(id_of_settimeout)`

- **window.setInterval()**: Ejecuta una función cada cierto intervalo de ms.

`setInterval(function, milliseconds, param1, param2, ...)`

- **windows.clearInterval()**: Cancela la ejecución de un setInterval.

`clearInterval(var)`

Propiedades	Descripción
hash	http://www.example.com/test.htm#part2
host	protocolo://maquina_host[:puerto]/camino_al_recurso
hostname	protocolo://maquina_host[:puerto]/camino_al_recurso
href	protocolo://maquina_host[:puerto]/camino_al_recurso
pathname	protocolo://maquina_host[:puerto]/camino_al_recurso
port	protocolo://maquina_host[:puerto]/camino_al_recurso
protocol	protocolo://maquina_host[:puerto]/camino_al_recurso
search	https://www.w3schools.com/submit.htm?email=someone@example.com
origin	protocolo://maquina_host[:puerto]/camino_al_recurso

Métodos	Descripción
<code>assign(url)</code>	Carga un nuevo documento. Este método hace que la ventana cargue y muestre el documento de la URL especificada por parámetro.
<code>reload(forcedReload)</code>	Este método realiza una recarga del recurso de la URL actual Parámetro optativo (true → recarga desde el servidor, false → recarga desde la caché)
<code>replace(url)</code>	Este método reemplaza el recurso actual ("desaparece" del historial) por el recurso de la URL pasada por parámetro.

Encontrar números en una cadena de texto:

```
const texto = "Hay 5 gatos, 10 perros y 2 pájaros.";
const numeros = texto.match(/\d+/g);
console.log(numeros); // ["5", "10", "2"]
```

La expresión regular `\d+` busca cualquier secuencia de dígitos (uno o más) en la cadena. `g` al final de la expresión regular significa "global", es decir, buscará todas las coincidencias en la cadena, no solo la primera.

Validar un código postal (5 dígitos):

```
const codigoPostal = "28001";
const esValido = /^\\d{5}$/.test(codigoPostal);
console.log(esValido); // true
```

`^` asegura que la cadena empiece desde el principio.

`\\d{5}` garantiza que haya exactamente 5 dígitos.

`$` asegura que la cadena termine justo después de los 5 dígitos.

Extraer todas las direcciones de correo electrónico de un texto

```
const texto = "Contact us at support@example.com or admin@domain.org";
const correos = texto.match(/\\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\. [A-Za-z]{2,}\\b/g);
console.log(correos); // ["support@example.com", "admin@domain.org"]
```

`\\b` asegura que las coincidencias estén delimitadas por espacios o caracteres no alfanuméricos (por ejemplo, puntos).

`[A-Za-z0-9._%+-]+` coincide con los caracteres válidos antes del `@`.

`@ [A-Za-z0-9.-]+` coincide con el dominio del correo.

`\\. [A-Za-z]{2, }` asegura que el dominio termine con un punto seguido de al menos dos letras (para cubrir las extensiones de dominio como `.com`, `.org`, etc.).

Validar direcciones IP en formato IPv4

```
const ip = "192.168.0.1";
const esValida =
/^((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$/i.test(ip);
console.log(esValida); // true
```

25[0-5] permite los números de 250 a 255.

2[0-4][0-9] permite los números de 200 a 249.

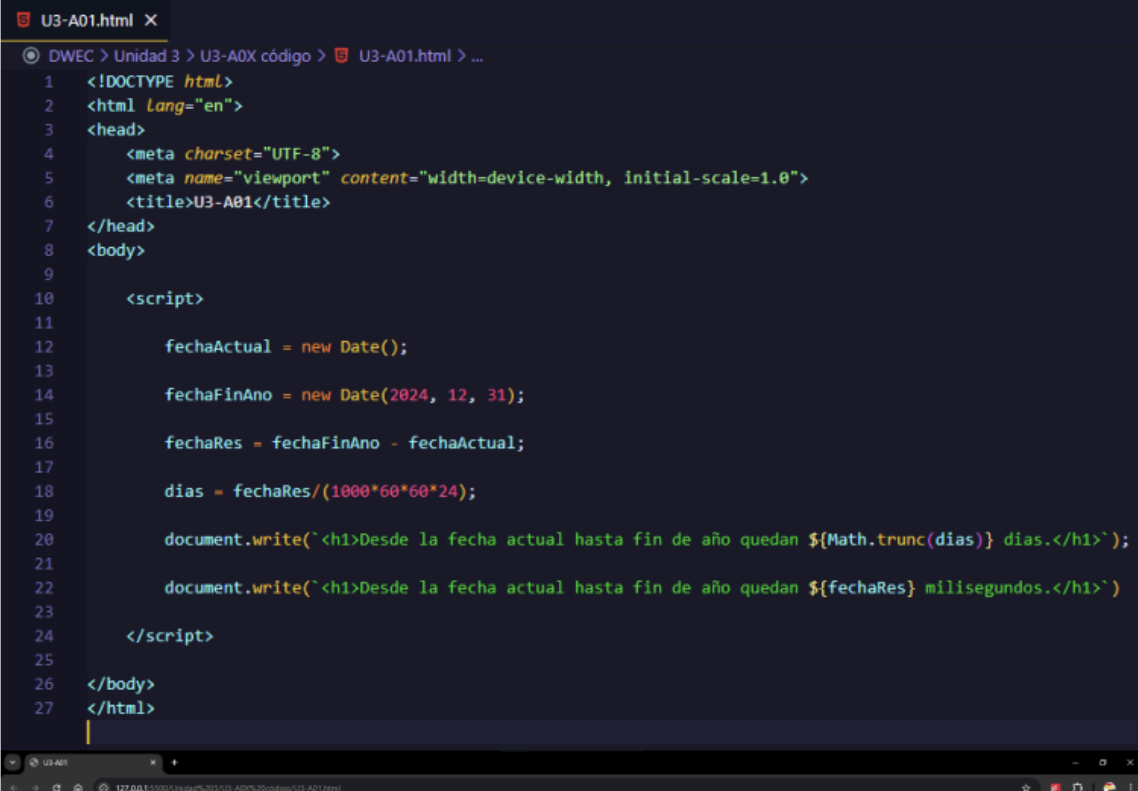
[01]?[0-9][0-9]? permite los números de 0 a 199.

{3} asegura que la expresión anterior se repita tres veces, para cubrir los tres primeros bloques de la IP.

El último bloque es el mismo que los anteriores, pero no tiene el punto final.

Calcular el tiempo restante expresado en días entre la fecha actual y la fecha de fin de año. La información deberá aparecer en la página web.

Calcularla también expresada en milisegundos.



```
U3-A01.html X
DWECC > Unidad 3 > U3-A0X código > U3-A01.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>U3-A01</title>
7 </head>
8 <body>
9
10  <script>
11
12    fechaActual = new Date();
13
14    fechaFinAño = new Date(2024, 12, 31);
15
16    fechaRes = fechaFinAño - fechaActual;
17
18    dias = fechaRes/(1000*60*60*24);
19
20    document.write(`<h1>Desde la fecha actual hasta fin de año quedan ${Math.trunc(dias)} dias.</h1>`);
21
22    document.write(`<h1>Desde la fecha actual hasta fin de año quedan ${fechaRes} milisegundos.</h1>`);
23
24  </script>
25
26 </body>
27 </html>
```

Desde la fecha actual hasta fin de año quedan 97 días.

Desde la fecha actual hasta fin de año quedan 8413651165 milisegundos.

Haciendo uso de los métodos vistos en clase e introduciendo tu fecha de nacimiento:

Mostrar en la página web : los meses, días y años que tienes a fecha actual.

```
U3-A03.html X
DWECC > Unidad 3 > U3-A0X código > U3-A03.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>U3-A03</title>
7  </head>
8  <body>
9
10     <script>
11
12         fechaNacimiento = new Date(2000, 11, 21);
13
14         fechaActual = new Date();
15
16         fechaRes = fechaActual.getTime() - fechaNacimiento.getTime();
17
18         meses = Math.trunc(fechaRes/(1000*60*60*24*31));
19
20         días = Math.trunc(fechaRes/(1000*60*60*24));
21
22         años = Math.trunc(fechaRes/(1000*60*60*24*365));
23
24         document.write(`<h1>Tengo ${años} años.</h1>`);
25         document.write(`<h1>Tengo ${meses} meses.</h1>`);
26         document.write(`<h1>Tengo ${días} días.</h1>`);
27
28     </script>
29
30 </body>
31 </html>
```

Tengo 23 años.

Tengo 280 meses.

Tengo 8709 días.

El usuario deberá introducir una hora y se deberá validar el ingreso de una hora con el formato hh:mm:ss o hh:mm

`/^([01]?[0-9]|2[0-3]):([0-5]?[0-9]):([0-5]?[0-9])?$/`

`([01]?[0-9]|2[0-3])`: Esto valida las horas (**hh**). Puede ser un número de dos dígitos, donde:

- `[01]?[0-9]` permite horas de **00** a **19**.
- `2[0-3]` permite horas de **20** a **23**.

`([0-5]?[0-9])`: Esto valida los minutos (**mm**) y segundos (**ss**), permitiendo números de **00** a **59**.

`(:[0-5]?[0-9])?`: Esto permite que los segundos sean opcionales. Si se incluyen, deben estar entre **00** y **59**.

Los `^` y `$` son anclas que aseguran que la cadena esté completamente alineada con el formato (inicio y fin de la cadena).

Valida una matrícula moderna (Formato: 0000 AAA) (incluido el espacio)

`/^\d{4} [A-Z]{3}$/`

`^\d{4}`:

- `^` indica el inicio de la cadena.
- `\d{4}` busca exactamente cuatro dígitos (del **0** al **9**).

(espacio):

- El espacio es obligatorio entre los números y las letras.

`[A-Z]{3}`:

- `[A-Z]` coincide con cualquier letra mayúscula del alfabeto (**A** a **Z**).
- `{3}` asegura que haya exactamente tres letras.

`$`:

- El símbolo `$` indica el final de la cadena, asegurando que no haya más caracteres después de la matrícula.

Confecciona una expresión regular que valide el ingreso del nombre de un día de la semana y un número de 1 o 2 dígitos

(ejemplo: lunes 31) - Expresión válida

(ejemplo: llunes 2) - Expresión no válida

```
/^(lunes|martes|miércoles|jueves|viernes|sábado|domingo) \d{1,2}$/
```

`^(lunes|martes|miércoles|jueves|viernes|sábado|domingo):`

- `^` asegura que la cadena comience con uno de los días de la semana.
- El operador `|` se usa para separar las diferentes opciones (de lunes a domingo).
- Los días de la semana están escritos en minúsculas exactamente como se indican, con las tildes y todo.

(espacio):

- Se asegura de que haya un espacio entre el nombre del día y el número que sigue.

`\d{1,2}:`

- `\d` significa que estamos buscando un dígito.
- `{1,2}` asegura que haya entre 1 y 2 dígitos numéricos, lo que cubre los números de 1 a 99.

`$:`

- Asegura que no haya más caracteres después del número, es decir, que la cadena termine ahí.

Crear una página principal que permita al usuario elegir entre dos operaciones matemáticas: suma o resta. Al seleccionar una de estas operaciones, se abrirá una nueva "pantalla" (una página en blanco) que pedirá al usuario que ingrese dos números, los cuales serán sumados o restados. Finalmente, el resultado se mostrará en otra "pantalla" utilizando document.write()

- Crea una página principal (index.html) con dos botones: "Sumar" y "Restar".
- Cuando el usuario haga clic en uno de los botones, se abrirá una nueva ventana (otra página en blanco) pidiéndole que ingrese dos números.
- Al ingresar los números, y al presionar un botón de "Calcular", se abrirá una tercera "pantalla" que mostrará el resultado de la operación.

```
index.html M x  sumar.html M  restar.html U  resultado.html M
DWECC > Unidad 3 > U3-A23 > index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>U3-A23</title>
7  </head>
8  <body>
9      <br><br>
10     <button onclick="window.open('sumar.html')">Sumar</button>
11     <button onclick="window.open('restar.html')">Restar</button>
12 </body>
13 </html>

index.html M  sumar.html M x  restar.html U  resultado.html M
DWECC > Unidad 3 > U3-A23 > sumar.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Sumar</title>
7  </head>
8  <body>
9      <form>
10         <label>
11             Primer operando
12             <input type="number" id="op1">
13         </label>
14         <br><br>
15         <label>
16             Segundo operando
17             <input type="number" id="op2">
18         </label>
19         <br><br>
20         <button onclick="imprimirResultado()">Calcular</button>
21     </form>
22     <script>
23         function imprimirResultado(){
24             res = parseInt(document.getElementById('op1').value) + parseInt(document.getElementById('op2').value);
25             resultado = window.open('resultado.html?res=${res}', "Resultado");
26         }
27     </script>
28 </body>
29 </html>
```

```
index.html M  sumar.html M  restar.html U X  resultado.html M
DWECC > Unidad 3 > U3-A23 > restar.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Restar</title>
7  </head>
8  <body>
9      <form>
10         <label>
11             Primer operando
12             <input type="number" id="op1">
13         </label>
14         <br><br>
15         <label>
16             Segundo operando
17             <input type="number" id="op2">
18         </label>
19         <br><br>
20         <button onclick="imprimirResultado()">Calcular</button>
21     </form>
22     <script>
23         function imprimirResultado(){
24             res = parseInt(document.getElementById('op1').value) - parseInt(document.getElementById('op2').value);
25             resultado = window.open('resultado.html?res=${res}', "Resultado");
26         }
27     </script>
28 </body>
29 </html>
```

```
index.html M  sumar.html M  restar.html U  resultado.html M X
DWECC > Unidad 3 > U3-A23 > resultado.html > html > body > script
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Resultado</title>
7  </head>
8  <body>
9      <script>
10         /* location es un objeto que representa la URL de la página actual y search accede
11         a la cadena de consulta de la URL, es decir, a partir del ? de la URL */
12         res = window.location.search;
13         // Convierte la cadena de parámetros en un objeto
14         res = new URLSearchParams(res);
15         // Obtengo el valor del parámetro "res" del objeto anterior
16         res = res.get("res");
17         // Imprimo el valor de res
18         document.write(`<h1>El resultado es: ${res}</h1>`);
19     </script>
20 </body>
21 </html>
```