



4. Cadenas de Texto (Strings)

String — Cadena de caracteres - JavaScript | MDN

El objeto String se utiliza para representar y manipular una secuencia de caracteres.

 https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/String

 mdn web docs

Las cadenas de texto, o **strings**, son uno de los tipos de datos más utilizados en JavaScript. Representan texto y son esenciales para cualquier aplicación que requiera manipular información textual, como nombres, mensajes, direcciones URL, entre otros. En este capítulo, aprenderás cómo crear, manipular y utilizar métodos básicos de los strings, además de explorar las potentes **template strings**.

Creación de Variables String

En JavaScript, puedes crear strings utilizando dos enfoques principales: **notación literal** y **constructor**.

Creación con Notación Literal

La forma más común de crear un string es usando comillas simples (`'`) o dobles (`"`). Esta es la opción preferida en la mayoría de los casos debido a su simplicidad.

Ejemplo:

```
let saludo = "Hola, mundo";  
let nombre = 'Juan';
```

Creación con el Constructor `new String()`

Aunque menos común, también puedes crear un string utilizando el constructor `String`. Sin embargo, esto crea un objeto de tipo `String` en lugar de un valor primitivo, lo que puede llevar a comportamientos inesperados al comparar valores.

Ejemplo:

```
let mensaje = new String("Hola");  
console.log(typeof mensaje); // "object"
```

Nota: Es recomendable usar la notación literal (`'` o `"`) en lugar del constructor, ya que los objetos String pueden complicar operaciones como comparaciones.

Propiedad `length` : Obtener la Longitud de un String

La propiedad `length` devuelve el número de caracteres en un string, incluyendo espacios y signos de puntuación.

Ejemplo:

```
let frase = "JavaScript es divertido";
console.log(frase.length); // 22
```

Esto es útil para verificar la longitud de entradas de usuario, validar campos de texto o recorrer caracteres individuales.

Concatenación de Strings

La concatenación permite combinar múltiples strings en uno solo. Hay varias formas de hacerlo:

1. Usando el Operador `+`:

```
let nombre = "Ana";
let saludo = "Hola, " + nombre + "!";
console.log(saludo); // "Hola, Ana!"
```

2. Usando el Método `concat()`:

```
let saludo = "Hola, ".concat(nombre, "!");
console.log(saludo); // "Hola, Ana!"
```

3. Usando Template Strings (ver más adelante):

```
let saludo = `Hola, ${nombre}!`;
console.log(saludo); // "Hola, Ana!"
```

Métodos Básicos de los Strings

JavaScript proporciona una amplia variedad de métodos para manipular strings. A continuación, exploraremos algunos de los más útiles:

`toUpperCase()` y `toLowerCase()`

Estos métodos convierten todos los caracteres de un string a mayúsculas o minúsculas, respectivamente.

Ejemplo:

```
let texto = "JavaScript";
console.log(texto.toUpperCase()); // "JAVASCRIPT"
console.log(texto.toLowerCase()); // "javascript"
```

`includes()`

Este método verifica si un string contiene una subcadena específica. Devuelve `true` si la encuentra, y `false` en caso contrario.

Ejemplo:

```
let frase = "El lenguaje JavaScript es poderoso";
console.log(frase.includes("JavaScript")); // true
console.log(frase.includes("Python")); // false
```

trim()

Elimina los espacios en blanco al inicio y al final de un string. Es útil para limpiar entradas de usuario.

Ejemplo:

```
let entrada = "  Hola, mundo  ";
console.log(entrada.trim()); // "Hola, mundo"
```

split()

Divide un string en un array de subcadenas basadas en un separador especificado.

Ejemplo:

```
let frase = "manzana,plátano,uva";
let frutas = frase.split(",");
console.log(frutas); // ["manzana", "plátano", "uva"]
```

Otros métodos útiles incluyen:

- `charAt(index)` : Obtiene el carácter en una posición específica.
- `indexOf(substring)` : Devuelve la posición de la primera aparición de una subcadena.
- `replace(oldValue, newValue)` : Reemplaza una parte del string por otra.

Template Strings: Sintaxis Moderna para Strings

Las **template strings** (también conocidas como literales de plantilla) son una característica introducida en ES6 que facilita la creación de strings complejos. Se escriben entre backticks (``) y permiten incluir expresiones dinámicas dentro del texto usando `${}`.

Ventajas de las Template Strings

1. Interpolación de Variables:

Puedes insertar variables directamente en el string sin necesidad de concatenación.

Ejemplo:

```
let nombre = "María";
let edad = 25;
```

```
let mensaje = `Mi nombre es ${nombre} y tengo ${edad} años.`;  
console.log(mensaje); // "Mi nombre es María y tengo 25 años."
```

2. Multilínea:

Las template strings permiten escribir texto en varias líneas sin necesidad de usar

`\n`.

Ejemplo:

```
let poema = `  
En un lugar de la Mancha,  
de cuyo nombre no quiero acordarme...  
`;  
console.log(poema);
```

3. Expresiones Dinámicas:

Puedes incluir expresiones complejas dentro de

`${}`.

Ejemplo:

```
let precio = 100;  
let descuento = 20;  
let total = `El precio final es ${precio - descuento}.`;  
console.log(total); // "El precio final es $80."
```