# JavaScript String Methods

**Camper:** Jhoan Sebastián Diaz Balta

**Trainer:** Cristian Diaz

**Investigation:**

## Longitud de cadena de JavaScript

El length La propiedad devuelve la longitud de una cadena:

**Ejemplo**

```
let text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
let length = text.length; // 26
```

## Extracción de caracteres de cadena

Hay 4 métodos para extraer caracteres de cadena:

- El Métodoat(*position*)
- El MétodocharAt(*position*)
- El MétodocharCodeAt(*position*)
- Uso del acceso a propiedades [] como en matrices

## Cadena de JavaScript charAt()

El charAt() El método devuelve el carácter en un valor especificado índice (posición) en una cadena:

Ejemplo

```
let text = "HELLO WORLD";
let char = text.charAt(0); // H
```

## Cadena de JavaScript charCodeAt()

El charCodeAt() El método devuelve el código del carácter en un índice específico en una cadena:

El método devuelve un código UTF-16 (un número entero entre 0 y 65535).

Ejemplo

```
let text = "HELLO WORLD";
let char = text.charCodeAt(0); // 72
```

## Código JavaScriptPointAt()

Ejemplos

Obtenga el valor del punto de código en la primera posición de una cadena:

```
let text = "HELLO WORLD";
let code = text.codePointAt(0);
```

## Cadena de JavaScript en at()

ES2022 introdujo el método de cadena at():

Ejemplos

Obtenga la tercera letra del nombre:

```
const name = "W3Schools";
let letter = name.at(2);
```

Obtenga la tercera letra del nombre:

```
const name = "W3Schools";
let letter = name[2];
```

El at() El método devuelve el carácter en un índice (posición) específico en una cadena.

El at() El método es compatible con todos los navegadores modernos desde marzo de 2022:

**Nota**

El at() El método es una nueva incorporación a JavaScript.

Permite el uso de índices negativos mientras charAt() no.

Ahora puedes usarlo myString.at(-2) en lugar de charAt(myString.length-2).

## Acceso a la propiedad [ ]

Ejemplo

```
let text = "HELLO WORLD";
let char = text[0];
```

Nota

El acceso a la propiedad puede ser un poco **impredecible:**

- Hace que las cadenas parezcan matrices (pero no lo son)
- Si no se encuentra ningún carácter, [ ] devuelve indefinido, mientras que charAt() devuelve una cadena vacía.
- Es de solo lectura. str[0] = "A" no da ningún error (¡pero no funciona!)

Ejemplo

```
let text = "HELLO WORLD";
text[0] = "A";    // Gives no error, but does not work
```

---

## Concatenación de cadenas de JavaScript()

concat()une dos o más cadenas:

**Ejemplo**

```
let text1 = "Hello";
let text2 = "World";
let text3 = text1.concat(" ", text2); // Hello World
```

El concat() Se puede utilizar el método en lugar del operador plus. Estas dos líneas hacen lo mismo:

**Ejemplo**

```
text = "Hello" + " " + "World!";
text = "Hello".concat(" ", "World!");
```

Nota

Todos los métodos de cadena devuelven una nueva cadena. No modifican la cadena original.

Formalmente dicho:

Las cadenas son inmutables: las cadenas no se pueden cambiar, solo reemplazar.

**Extracción de piezas de cuerdas**

Hay 3 métodos para extraer una parte de una cadena:

- slice(*start*, *end*)
- substring(*start*, *end*)
- substr(*start*, *length*)

**Segmento de cadena de JavaScript()**

slice()extrae una parte de una cadena y devuelve la parte extraída en una nueva cadena.

El método toma 2 parámetros: posición inicial y posición final (final no incluido).

**Ejemplo**

Corta una porción de una cuerda desde la posición 7 a la posición 13:

```
let text = "Apple, Banana, Kiwi";
        let part = text.slice(7, 13);
```

**Ejemplos**

Si omite el segundo parámetro, el método eliminará el resto de la cadena:

```
let text = "Apple, Banana, Kiwi";
let part = text.slice(7);
```

Si un parámetro es negativo, la posición se cuenta desde el final de la cadena:

```
let text = "Apple, Banana, Kiwi";
let part = text.slice(-12); // Banana, Kiwi
```

Este ejemplo corta una porción de una cadena desde la posición -12 a la posición -6:

```
let text = "Apple, Banana, Kiwi";
        let part = text.slice(-12, -6);
```

## Subcadena de cadena de JavaScript()

substring() es similar a slice().

La diferencia es que los valores iniciales y finales inferiores a 0 se tratan como 0 en substring().

Ejemplo

```
let str = "Apple, Banana, Kiwi";
let part = str.substring(7, 13);
```

Si omite el segundo parámetro, substring() cortará el resto de la cadena.

---

## Conversión a mayúsculas y minúsculas

Una cadena se convierte a mayúsculas con toUpperCase():

Una cadena se convierte a minúsculas con toLowerCase():

## Cadena de JavaScript toUpperCase()

Ejemplo

```
let text1 = "Hello World!";
let text2 = text1.toUpperCase();
```

## Cadena de JavaScript toLowerCase()

Ejemplo

```
let text1 = "Hello World!";       // String
let text2 = text1.toLowerCase();  // text2 is text1 converted to lower
```

## La cadena JavaScript está bien formada ()

El isWellFormed() el método devuelve true si una cuerda está bien formada.

De lo contrario regresa false.

Una cuerda no está bien formada si contiene **sustitutos solitarios**.

A **sustituto solitario** es un punto de código sustituto Unicode que no forma parte de un par sustituto válido Se utiliza para representar caracteres en codificación UTF-16.

Ejemplos

```
let text = "Hello world!";
let result = text.isWellFormed(); // True
```

```
let text = "Hello World \uD800";
let result = text.isWellFormed(); // False
```

## Cadena de JavaScript toWellFormed()

El método String toWellFormed() devuelve una nueva cadena donde todos los "sustitutos solitarios" se reemplazan con el carácter de reemplazo Unicode (U+FFFD).

Ejemplos

```
let text = "Hello World \uD800";
let result = text.toWellFormed();
```

---

## Recorte de cadena de JavaScript()

El trim() El método elimina los espacios en blanco de ambos lados de una cadena:

Ejemplo

```
let text1 = "      Hello World!      ";
let text2 = text1.trim();
```

## Cadena JavaScript trimStart()

ECMAScript 2019 Se agregó el método String trimStart() a JavaScript.

El trimStart() El método funciona así trim(), pero elimina los espacios en blanco sólo desde el comienzo de una cadena.

Ejemplo

```
let text1 = "     Hello World!     ";
let text2 = text1.trimStart();
```

## Cadena JavaScript trimEnd()

ECMAScript 2019 Se agregó el método de cadena trimEnd() a JavaScript.

El trimEnd() El método funciona así trim(), pero elimina los espacios en blanco sólo del final de una cadena.

Ejemplo

```
let text1 = "     Hello World!     ";
let text2 = text1.trimEnd();
```

---

## Relleno de cadenas de JavaScript

ECMAScript 2017 Se agregaron dos nuevos métodos de cadena a JavaScript: padStart() y padEnd() para soportar el relleno al principio y al final de una cadena.

## Cadena JavaScript padStart()

El padStart() El método rellena una cadena desde el principio.

Rellena una cuerda con otra cuerda (varias veces) hasta que alcanza una longitud determinada.

Ejemplos

Rellene una cadena con "0" hasta que alcance la longitud 4:

```
let text = "5";
let padded = text.padStart(4,"0");
```

Rellene una cadena con "x" hasta que alcance la longitud 4:

```
let text = "5";
let padded = text.padStart(4,"x");
```

Nota

El padStart() El método es un método de cadena.

Para rellenar un número, primero conviértalo en una cadena.

Vea el ejemplo a continuación.

Ejemplo

```
let numb = 5;
let text = numb.toString();
let padded = text.padStart(4,"0");
```

## Cadena JavaScript padEnd()

El padEnd() El método rellena una cadena desde el final.

Rellena una cuerda con otra cuerda (varias veces) hasta que alcanza una longitud determinada.

Ejemplos

```
let text = "5";
let padded = text.padEnd(4,"0");
```

```
let text = "5";
let padded = text.padEnd(4,"x");
```

```
let numb = 5;
let text = numb.toString();
let padded = text.padEnd(4,"0");
```

---

## Repetición de cadena de JavaScript()

El repeat() El método devuelve una cadena con varias copias de una cadena.

El repeat() El método devuelve una nueva cadena.

El repeat() El método no cambia la cadena original.

Ejemplos

Crea copias de un texto:

```
let text = "Hello world!";
let result = text.repeat(2);
```

```
let text = "Hello world!";
let result = text.repeat(4);
```

Sintaxis

```
string.repeat(count)
```

---

## Reemplazo de contenido de cadena

El replace() El método reemplaza un valor especificado con otro valor en una cadena:

Ejemplo

```
let text = "Please visit Microsoft!";
let newText = text.replace("Microsoft", "W3Schools");
```

Nota

El replace() El método no cambia la cadena en la que se llama.

El replace() El método devuelve una nueva cadena.

El replace() El método reemplaza **sólo el primero** partido

Si desea reemplazar todas las coincidencias, utilice una expresión regular con el indicador /g establecido. Vea los ejemplos a continuación.

Por defecto, el replace() El método reemplaza **sólo el primero** partido:

Ejemplo

```
let text = "Please visit Microsoft and Microsoft!";
let newText = text.replace("Microsoft", "W3Schools");
```

Por defecto, el replace() El método distingue entre mayúsculas y minúsculas. Escribiendo MICROSOFT (con mayúscula) no funcionará:

Ejemplo

```
let text = "Please visit Microsoft!";
let newText = text.replace("MICROSOFT", "W3Schools");
```

Para reemplazar casos que no distinguen entre mayúsculas y minúsculas, utilice a **expresión regular** con un /i bandera (insensible):

Ejemplo

```
let text = "Please visit Microsoft!";
let newText = text.replace(/MICROSOFT/i, "W3Schools");
```

Para reemplazar todas las coincidencias, utilice a **expresión regular** con a /g bandera (partido global):

Ejemplo

```
let text = "Please visit Microsoft and Microsoft!";
let newText = text.replace(/Microsoft/g, "W3Schools");
```

## Cadena JavaScript replaceAll()

En 2021, JavaScript introdujo el método de cadena replaceAll():

Ejemplo

```
text = text.replaceAll("Cats","Dogs");
text = text.replaceAll("cats","dogs");
```

El replaceAll() El método le permite especificar una expresión regular en lugar de una cadena a reemplazar.

Si el parámetro es una expresión regular, se debe establecer el indicador global (g); de lo contrario Se lanza un TypeError.

Ejemplo

```
text = text.replaceAll(/Cats/g,"Dogs");
text = text.replaceAll(/cats/g,"dogs");
```

---

## Convertir una cadena en una matriz

**Si desea trabajar con una cadena como matriz, puede convertirla en una matriz.**

## División de cadenas de JavaScript ()

Una cadena se puede convertir en una matriz con el split() método:

Ejemplo

```
text.split(",")    // Split on commas
text.split(" ")    // Split on spaces
text.split("|")    // Split on pipe
```

Si se omite el separador, la matriz devuelta contendrá la cadena completa en el índice [0].

Si el separador es "", la matriz devuelta será una matriz de uno solo Personajes:

Ejemplo

```
text.split("")
```