String.prototype.localeCompare()

Baseline Widely available

El método **`localeCompare()`** retorna un número indicando si una cadena de carateres de referencia va antes, después o si es la misma que la cadena dada en orden alfabético.

Pruébalo

JavaScript Demo: String.localeCompare()

```
const a = "réservé"; // With accents, lowercase
const b = "RESERVE"; // No accents, uppercase

console.log(a.localeCompare(b));
// Expected output: 1
console.log(a.localeCompare(b, "en", { sensitivity: "base" }));
// Expected output: 0

```

Los nuevos argumentos `locales` y `options` permiten a las aplicaciones especificar el idioma cuyo orden alfabético se debe usar y configurar el comportamiento de la función. En implementaciones antiguas, que ignoran los argumentos `locales` y `options`, la localización y el orden alfabético usados son completamente dependientes de la implementación.

Sintaxis

JS

```
localeCompare(compareString);
localeCompare(compareString, locales);
localeCompare(compareString, locales, options);
```

Parámetros

`compareString`

La cadena de caracteres contra la cual se compara la `referenceStr`.

`locales` y `options`

Estos argumentos configuran el comportamiento de la función y le permiten a las aplicaciones especificar el idioma cuyas convenciones de formato se deben usar. En implementaciones que ignoran los argumentos `locales` y `options`, la configuración regional usada y la forma de la cadena devuelta son completamente dependientes de la implementación.

Consulte el constructor de `Intl.Collator()` (inglés) para obtener detalles sobre estos parámetros y cómo usarlos.

Valor de retorno

Un número **negativo** si `referenceStr` ocurre antes de `compareString`; **positivo** si `referenceStr` ocurre después de `compareString`; `0` si son equivalentes.

Descripción

Retorna un entero que indica si la cadena `referenceStr` va antes, despues o si es equivalente a la cadena `compareString`.

- Negativo cuando `referenceStr` ocurre antes que `compareString`.
- Positivo cuando `referenceStr` ocurre después que `compareString`.
- Retorna `0` si son equivalentes.

> **Alerta:** No se debe asumir un valor de retorno exacto de `-1` o `1`.
>
> Resultados de enteros positivos y negativos pueden variar entre navegadores (al igual que entre versiones de navegadores) ya que la especificación del W3C solo requiere valores positivos y negativos. Algunos

> navegadores podrían retornar `-2` o `2`, o incluso otro valor positivo o negativo.

Rendimiento

Cuando se compara un gran número de cadenas, como cuando se ordenan arreglos de gran tamaño, es mejor crear un objeto `Intl.Collator` y usar la función provista por su propiedad `compare`.

Ejemplos

Uso de `localeCompare()`

JS

```
// La letra "a" va antes que "c" por lo que entrega un valor negativo
"a".localeCompare("c"); // -2 o -1 (o cualquier otro valor negativo)

// Alfabéticamente la palabra "check" va después que "against" por lo que resulta
// en un valor positivo.
"check".localeCompare("against"); // 2 o 1 (u otro valor positivo)

// "a" y "a" son equivalentes por lo que resulta en un valor neutral de cero.
"a".localeCompare("a"); // 0
```

Ordenar un arreglo

`localeCompare()` permite ordenar un arreglo independientemente de mayúsculas y minúsculas.

JS

```
let items = ["réservé", "Premier", "Cliché", "communiqué", "café", "Adieu"];
items.sort((a, b) => a.localeCompare(b, "fr", { ignorePunctuation: true }));
// ['Adieu', 'café', 'Cliché', 'communiqué', 'Premier', 'réservé']
```

Determinar soporte del navegador para los argumentos extendidos

Los argumentos `locales` y `options` no están soportados en todos los navegadores aún.

Para determinar si una implementación los soporta, usa el argumento `"i"` (un requerimiento de que las etiquetas de lenguaje ilegales sean rechazadas) y verifica si se lanza una excepción `RangeError`:

JS

```
function localeCompareSupportsLocales() {
  try {
    "foo".localeCompare("bar", "i");
  } catch (e) {
    return e.name === "RangeError";
  }
  return false;
}
```

Uso de `locales`

Los resultados provistos por `localeCompare()` varían por cada lenguaje. Para obtener el orden del lenguaje usado en la interfaz de usuario de tu aplicación, se debe asegurar de especificar dicho lenguaje (y posiblemente algunos lenguajes por defecto) usando el argumento `locales`:

JS

```
console.log("ä".localeCompare("z", "de")); // un valor negativo: en alemán, ä se ordena antes que z
console.log("ä".localeCompare("z", "sv")); // un valor positivo: en sueco, ä se ordena después que z
```

Uso de `options`

Los resultados provistos por `localeCompare()` se pueden personalizar usando el argumento `options`:

JS

```
// en alemán, ä tiene a a como letra base
console.log("ä".localeCompare("a", "de", { sensitivity: "base" })); // 0

// en sueco, ä y a son letras base separadas
console.log("ä".localeCompare("a", "sv", { sensitivity: "base" })); // un valor positivo
```

Ordenamiento numérico

JS

```
// por defecto, "2" > "10"
console.log("2".localeCompare("10")); // 1
```

```
// numérico usando options:
console.log("2".localeCompare("10", undefined, { numeric: true })); // -1

// numérico usando la etiqueta de locales:
console.log("2".localeCompare("10", "en-u-kn-true")); // -1
```

Especificaciones

Specification
ECMAScript® 2026 Language Specification # sec-string.prototype.localecompare
ECMAScript® 2026 Internationalization API Specification # sup-String.prototype.localeCompare

Compatibilidad con navegadores

Report problems with this compatibility data • View data on GitHub

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android	WebView on iOS	Bun	Deno	Node.js
`localeCompare`	✓ 1	✓ 12	✓ 1	✓ 7	✓ 3	✓ 18	✓ 4	✓ 10.1	✓ 1	✓ 1	✓ 4.4	✓ 1	✓ 1	✓ 1	✓ 0.10
`locales` parameter	✓ 24	✓ 12	✓ 29	✓ 15	✓ 10	✓ 26	✓ 56	⊗ No	✓ 10	✓ 1.5	⊗ No	✓ 10	⊗ No	✓ 1.8 …	✓ 13 …
`options` parameter	✓ 24	✓ 12	✓ 29	✓ 15	✓ 10	✓ 26	✓ 56	⊗ No	✓ 10	✓ 1.5	⊗ No	✓ 10	⊗ No	✓ 1	✓ 0.12

✓ Full support ⊘ Partial support ⊗ No support … Has more compatibility info.

Véase también

- `Intl.Collator`

mdn_

Your blueprint for a better internet.