Tema 3 Objetos Nativos en Javascript

Practica 03-01b String *template literals bacticks o comillas magicas*

Nombre		Curso	
Apellidos		Fecha	

Las comillas invertidas ` (también llamadas *backticks*) en **JavaScript** se utilizan para crear **template literals** o **plantillas de texto**, que son una forma más flexible y potente de manejar cadenas de texto que las comillas simples ' o dobles ".

Las comillas invertidas ` son útiles para:

- Crear **plantillas HTML dinámicas**.
- Escribir **texto multilínea** sin escapes.
- **Interpolar variables y expresiones** fácilmente.
- Mejorar la **legibilidad** del código

1. Sintaxis

```
let mensaje = `Hola mundo`;
console.log(mensaje); // Hola mundo
```

A simple vista parece igual que usar comillas normales, pero las *template literals* ofrecen **muchas ventajas**.

2. Interpolación de variables y expresiones

La principal ventaja es que puedes **insertar variables o expresiones dentro del texto** usando la sintaxis ${ }.

```
let nombre = "Ana";
let edad = 25;
let mensaje = `Hola, mi nombre es ${nombre} y tengo ${edad} años.`;
console.log(mensaje);
// Hola, mi nombre es Ana y tengo 25 años.
```

Incluso puedes incluir **expresiones completas** dentro de ${}:

```
let a = 5;
let b = 3;
console.log(`La suma de ${a} + ${b} es ${a + b}`);
// La suma de 5 + 3 es 8
```

3. Cadenas multilínea (sin usar \n)

Con las comillas normales, si querías escribir varias líneas, tenías que usar \n:

```
let texto = "Línea 1\nLínea 2";
```

Con backticks, puedes escribir directamente varias líneas:

```
let texto = `Línea 1
Línea 2
Línea 3`;
console.log(texto);
/*
Línea 1
Línea 2
Línea 3
*/
```

Esto es muy útil para generar **HTML** o **textos largos**:

```
let html = `
<div>
  <h1>Título</h1>
  <p>Esto es un párrafo.</p>
</div>`;
```

4. Funciones *tagged templates* (uso avanzado)

También puedes usar funciones llamadas *tags* para procesar las cadenas antes de que se formen. Por ejemplo:

```
function ejemplo(strings, ...values) {
  console.log(strings); // partes literales del texto
  console.log(values);  // los valores interpolados
}

let nombre = "Carlos";
let edad = 30;

ejemplo`Hola ${nombre}, tienes ${edad} años.`;
// strings → ["Hola ", ", tienes ", " años."]
// values → ["Carlos", 30]
```

Esto se usa en librerías como **styled-components** (React) o para crear **plantillas personalizadas**.

Tipo de comillas	Permite variables ${}	Permite multilínea	Ejemplo
'comillas simples'	NO	NO	'Hola'
"comillas dobles"	NO	NO	"Hola"
`backticks`	SI	SI	`Hola ${nombre}`

6. Ejemplo1: Generar una tarjeta de usuario en HTML

Usamos las comillas invertidas (`) para **generar HTML dinámico** con variables e interpolación . Supón que tenemos datos de un usuario en variables JavaScript y queremos crear una tarjeta con su información.

```
// Datos del usuario
const nombre = "Laura García";
const edad = 29;
const ciudad = "Madrid";
const intereses = ["fotografía", "viajes", "tecnología"];

// Creamos una plantilla HTML usando backticks
const tarjetaUsuario = `
  <div class="tarjeta">
    <h2>${nombre}</h2>
    <p>Edad: ${edad}</p>
    <p>Ciudad: ${ciudad}</p>
    <h4>Intereses:</h4>
    <ul>
      ${intereses.map(interes =>
`<li>${interes}</li>`).join("")}
    </ul>
  </div>
`;

// Mostramos el resultado en consola (o en una página)
document.writeln(tarjetaUsuario);
```

7. Explicación

1. **Variables dinámicas**:
   Se insertan directamente con ${nombre}, ${edad}, ${ciudad}.
   Esto evita concatenar texto con el operador +, que era más engorroso.

   ```
   // Forma antigua
   const mensaje = "Hola, " + nombre + ". Tienes " + edad + " años.";
   ```

   Con backticks:

   ```
   const mensaje = `Hola, ${nombre}. Tienes ${edad} años.`;
   ```

2. **Multilínea real**:
   Observa que el HTML tiene saltos de línea y sangrías sin necesidad de usar \n.

3. **Inserción de listas con expresiones**:
   Usamos ${intereses.map(...)} para recorrer el array y generar varias etiquetas <li>.
   El método .join("") une todos los elementos sin comas.

   ```
   ${intereses.map(interes => `<li>${interes}</li>`).join("")}
   ```

Práctica

Ejercicio 1. Copia el ejemplo1 y modificalo para que tengas tres usuarios distintos (como no hemos visto aun arrays solo crea tres variables distintas)

Ejercicio 2. **Formatea una descripción con datos dinámicos** Crear un texto descriptivo usando backticks e interpolación, luego usar funciones de String para manipularlo.

```
// Ejemplo de entrada
const producto = "auriculares inalámbricos";
const precio = 59.99;
const marca = "SoundMax";

// Tareas:
// Usa backticks para crear un texto como:
// "Los Auriculares Inalámbricos de la marca SoundMax cuestan 59.99€."
// Convierte la primera letra de cada palabra a mayúscula (usa toUpperCase, split, map, join).
//  Cambia el símbolo € por "euros" (usa replace).
```

Ejercicio 3. **Construiye una plantilla de correo** Crear una plantilla de email personalizada y limpiar texto con métodos de String.

```
// Datos
const nombre = "   juan pérez   ";
const producto = "Teclado mecánico RGB";
const fecha = "09/10/2025";

// Tareas:
//  Quita espacios extra en el nombre (usa trim()).
// Crea una plantilla de correo con backticks:
//    "Hola Juan Pérez, tu pedido del producto Teclado mecánico RGB se enviará el 09/10/2025."
// Convierte la primera letra del nombre y apellido en mayúsculas (usa split y map).
// Convierte todo el mensaje a mayúsculas (usa toUpperCase).
```

Ejercicio 4. **Genera una tabla de datos en HTML.** Usar template literals y métodos de cadena para generar un bloque HTML limpio.

```
// Datos
const usuarios = ["   ana ", "PEDRO", "maria   "];

// Tareas:
// Limpia los nombres (trim() y toLowerCase()).
// Capitaliza la primera letra de cada uno.
// Usa backticks para crear un bloque HTML con una lista <ul> donde cada <li> tenga un nombre formateado.
// (Extra) Usa padEnd() o padStart() para alinear los nombres si los imprimes en consola.
```

Ejercicio 5. **Análisis de texto dinámico.** Analizar un texto generado con template literals y aplicar varias funciones de String.

```
// Datos
const nombre = "Lucía";
const hobby = "leer libros de ciencia ficción";

// Tareas:
//  Crea un mensaje: `A ${nombre} le encanta ${hobby}.`
```


```
//  Muestra cuántas letras tiene el mensaje (length).
//  Verifica si el texto incluye la palabra “ficción” (includes()).
//  Reemplaza “ficción” por “aventuras” (replace()).
//  Convierte la oración a minúsculas y luego a mayúsculas (toLowerCase(), toUpperCase()).

// Convierte todo el mensaje a mayúsculas (usa toUpperCase).
```

Ejercicio 6. **Plantilla de perfil resumido.** Generar un resumen de usuario y recortar texto si es demasiado largo.

```
const usuario = {
  nombre: "Beatriz",
  descripcion: "Apasionada del desarrollo web, la inteligencia artificial y los videojuegos."
};

// Tareas:
//  Genera un texto con backticks: `Perfil de ${usuario.nombre}: ${usuario.descripcion}`
// Si la descripción tiene más de 50 caracteres, recórtala y añade "..." al final (usa slice y length).
// Muestra el resultado final.
```