

# Template strings

**DigitalHouse** >  
Coding School



**Certified Tech  
Developer**  
The Ultimate Degree

# Índice

1. [Sintaxis](#)
2. [Ejemplos de uso](#)
3. [Modificar el HTML](#)

# 1 | Sintaxis

# Sintaxis de un template string

## Texto/String

Este valor se mantendrá constante.

## Código JavaScript

Este código se ejecutará cada vez que el template string sea utilizado.



The diagram shows a code editor with a dark background. On the left, a label 'JS' is in a dark blue box. The code is ``Mi variable vale ${miVariable}``. A green bracket above the code spans from the first backtick to the end of the text 'Mi variable vale'. A grey bracket above the code spans from the start of the interpolation `\${miVariable}` to the closing backtick. Below the code, an orange bracket is under 'Mi variable vale', and two red brackets are under the `\${miVariable}` interpolation.

```
JS `Mi variable vale ${miVariable}`
```

## Template String

Inician con comillas cruzadas, si intentan utilizar las comillas simples ( ' ) o dobles ( " ), JavaScript solo interpretará que es un texto.

## Apertura de código

Con estas llaves indicamos que lo envuelto será interpretado como código JavaScript, es decir, si `miVariable` tiene un valor de 14, la salida de este template sería: **Mi variable vale 14.**

## 2 | Ejemplos de uso

# Ejemplo 1

```
const nombre= 'Mauro';  
const miTemplate = `Mi nombre es ${nombre}`;  
console.log(miTemplate);
```

El resultado sería:  
**Mi nombre es mauro**



Recordemos probar este código en nuestra computadora.

## Ejemplo 2

```
const miTemplate = `2 * 3 es ${ 2 * 3 }`;  
console.log(miTemplate);
```

El resultado sería:  
**2 \* 3 es 6**

## Ejemplo 3

```
function suma(var1, var2) {  
    return var1 + var2  
}  
  
const temp = `El resultado es ${suma(2,2)}`;  
console.log(temp);
```

El resultado sería:  
**El resultado es 4**



Como se puede ver, dentro de las llaves se puede ejecutar cualquier código JavaScript, pero es buena costumbre que sea lo más sencillo posible



# 3 | Modificar HTML

# Contenido dinámico

El módulo pasado vimos dos propiedades muy útiles para modificar nuestro documento HTML mediante JavaScript: **innerText** e **innerHTML**, ahora podemos probar el siguiente fragmento de código en tu computadora antes de continuar:

```
const template = `

# 

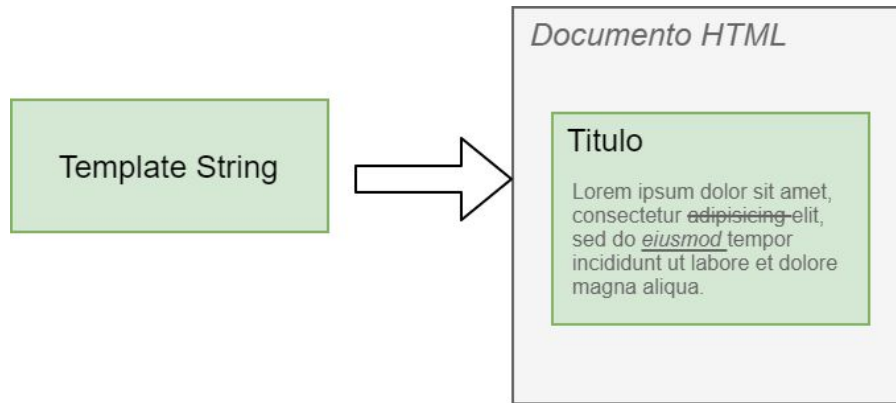

```



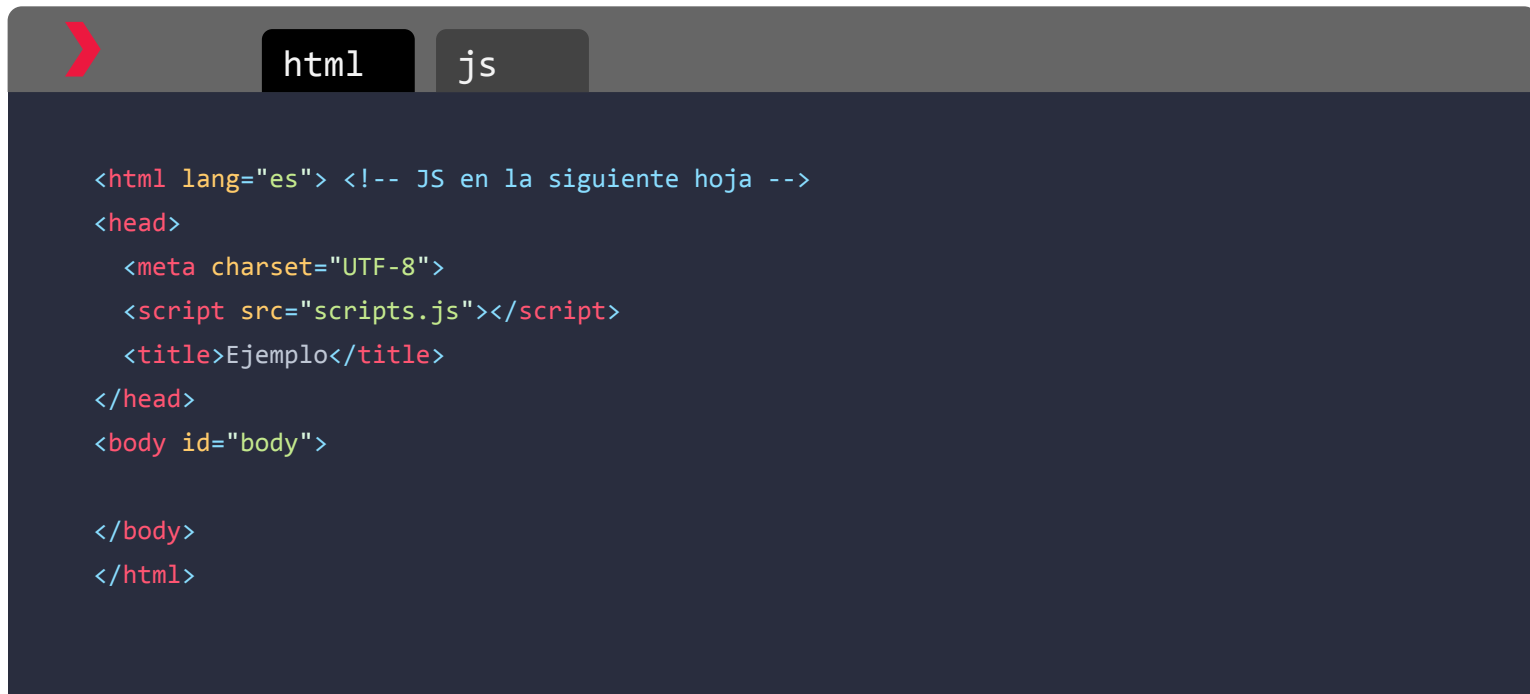
¡No olvidemos enlazar nuestro script al HTML!

# InnerHTML

A diferencia de `innerText`, `innerHTML` permite incluir etiquetas en nuestro template string y que estas sean interpretadas correctamente por el navegador, esto nos trae la posibilidad de agregar muchísimo dinamismo a nuestras páginas.



# ¡Empecemos a probarlo!



```
<html lang="es"> <!-- JS en la siguiente hoja -->
<head>
  <meta charset="UTF-8">
  <script src="scripts.js"></script>
  <title>Ejemplo</title>
</head>
<body id="body">

</body>
</html>
```



html

js

```
function escribirHTML(titulo, texto) {  
  const body = document.getElementById('body');  
  const miTemplate = `  
    <h1>${titulo}</h1>  
    <p>${texto}</p>  
  `;  
  body.innerHTML += miTemplate;  
}  
  
escribirHTML('Hola', 'Esto es un ejemplo de template string en html.');
```

```
escribirHTML('Es dinámico', 'Podemos insertar elementos HTML mediante <b>JavaScript</b>');
```

```
escribirHTML('Facilita la programación', 'Evita escribir mucho código y reutilizar el que si escribamos.');
```

```
escribirHTML('En este ejemplo', 'Hemos utilizado una única función para poder escribir 4 veces en HTML, ¿te imaginás lo que sería esto sin esta función?');
```

# Template strings

Template strings, template literals, plantillas literales, entre otros, son todos nombres que recibe esta funcionalidad y es una de las bases de la programación dinámica en la web, de ahora en más iremos profundizando en diferentes herramientas y utilidades de JavaScript, y muy frecuentemente terminaremos utilizando templates.



Si tenemos alguna duda, consultemos con los profesores y ¡animémonos a practicar!

DigitalHouse>  
Coding School