



1º DAM/DAW EDE

---

U4. Depuración (debugging)

AP4 - Depuración (debugging)



## Ejercicio 1

### Descripción:

Sigue los pasos explicados en los enunciados para generar la documentación solicitada.

### Objetivo:

Aprender a depurar código y seguir una traza. Entender las ventajas del uso de depurador.

### Bibliografía:

Recursos didácticos de Florida Oberta U4.

### Actividad a realizar:

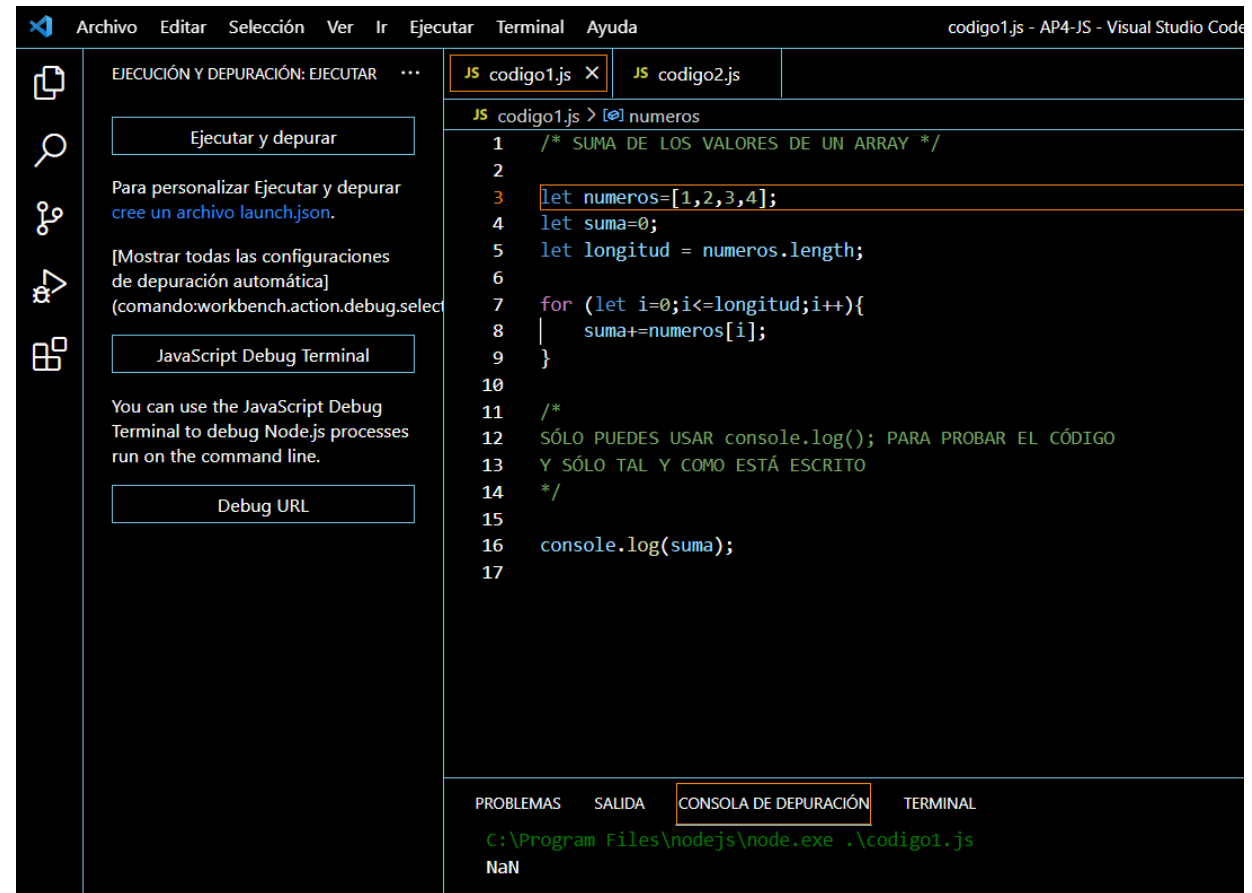
Cada enunciado va a plantear un caso con una serie de pasos a seguir. Cada uno los pasos, además de realizarse, deben quedar convenientemente documentado, mediante capturas de pantalla, redacción de texto explicativo, generación de tablas, y/o cualquier otro recurso que se considere conveniente.

### Pasos a seguir:

- 1 - Leer y analizar cada caso y cada paso dentro de cada caso.
- 2 - Ejecutar cada paso hasta completar todos los casos.
- 3 - Generar un **un documento PDF**, debidamente identificado, que incluya cada enunciado con la respuesta correspondiente.
- 4 - Entregar a través de Florida Oberta en los plazos indicados.

## Caso 1

- **Paso 1:** abre el archivo **codigo1.js** proporcionado, mediante Visual Studio Code. Ejecuta y depura el código. Comprueba que nos muestra un error: **'NaN'**.



The screenshot shows the Visual Studio Code interface with the file 'codigo1.js' open. The code in the editor is as follows:

```
1  /* SUMA DE LOS VALORES DE UN ARRAY */
2
3  let numeros=[1,2,3,4];
4  let suma=0;
5  let longitud = numeros.length;
6
7  for (let i=0;i<=longitud;i++){
8      suma+=numeros[i];
9  }
10
11 /*
12 SÓLO PUEDES USAR console.log(); PARA PROBAR EL CÓDIGO
13 Y SÓLO TAL Y COMO ESTÁ ESCRITO
14 */
15
16 console.log(suma);
17
```

The left sidebar shows the 'Ejecución y depuración' (Execution and Debugging) panel with the 'Ejecutar y depurar' (Run and Debug) button highlighted. The bottom status bar shows the 'CONSOLA DE DEPURACIÓN' (Debug Console) tab, which displays the output:

```
C:\Program Files\nodejs\node.exe .\codigo1.js
NaN
```

## Caso 1

- **Paso 2:** introduce un punto de interrupción para inspeccionar el valor de las variables '**i**' y '**suma**'. Elabora una traza. Es decir, anota en cada iteración el valor de estas variables en una tabla con esta estructura:

Iteración	i	suma
1	0	1
2	1	3
3	2	6
4	3	10
5	4	NaN

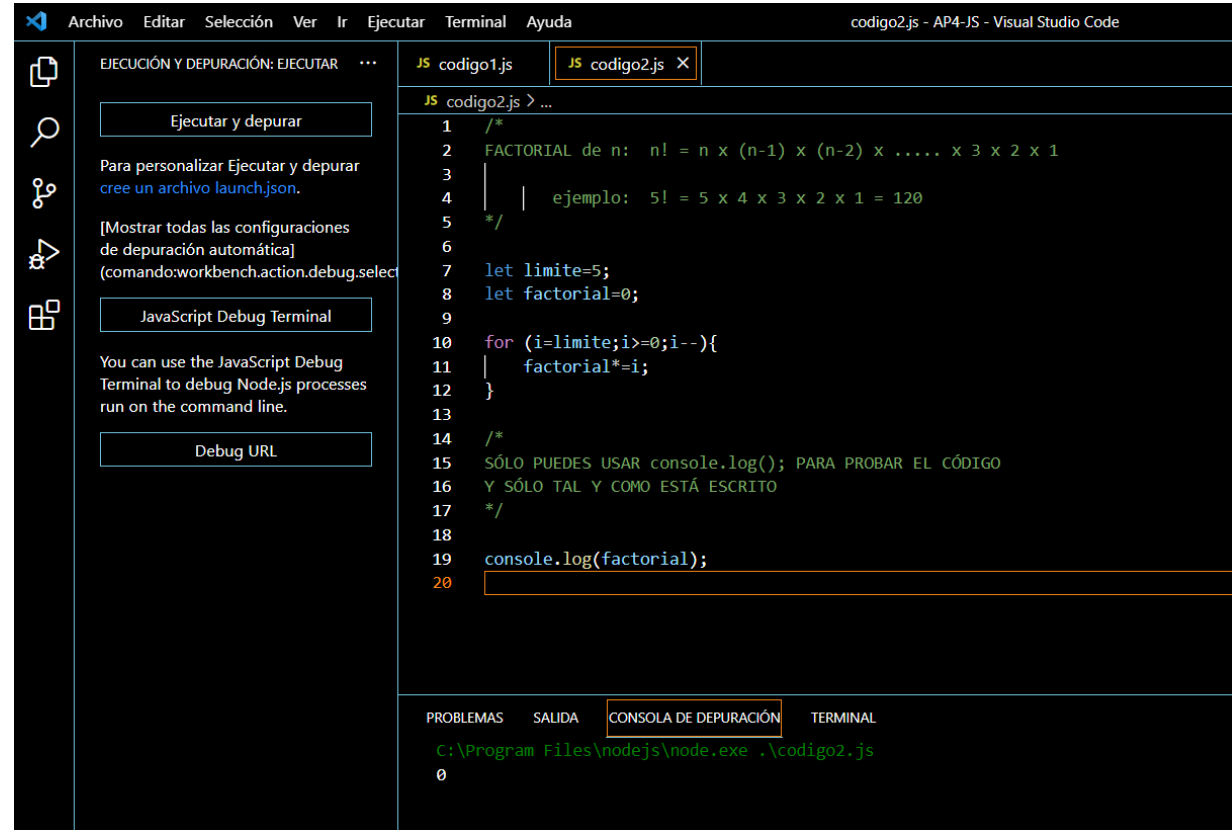
## Caso 1

- **Paso 3:** corrige el error en el código para que se pueda realizar la suma de todos los números y nos devuelva el resultado correcto. Explica en qué ha consistido la corrección e inspecciona de nuevo las variables para comprobar que es correcto.

```
for (let i=0;i<longitud;i++){  
    | suma+=numeros[i];  
}
```

## Caso 2

- **Paso 1:** abre el archivo **codigo2.js** proporcionado, mediante Visual Studio Code. Ejecuta y depura el código. Comprueba que, en este caso, nos muestra un cero: **'0'**. Ningún factorial puede ser 0 por definición, por lo que algo está pasando.



The screenshot shows the Visual Studio Code interface with the file 'codigo2.js' open. The code defines a factorial function and calculates the factorial of 5. The console output shows '0'.

```
1  /*
2  FACTORIAL de n:  n! = n x (n-1) x (n-2) x ..... x 3 x 2 x 1
3
4  | ejemplo:  5! = 5 x 4 x 3 x 2 x 1 = 120
5  */
6
7  let limite=5;
8  let factorial=0;
9
10 for (i=limite;i>=0;i--){
11   factorial*=i;
12 }
13
14 /*
15 SÓLO PUEDES USAR console.log(); PARA PROBAR EL CÓDIGO
16 Y SÓLO TAL Y COMO ESTÁ ESCRITO
17 */
18
19 console.log(factorial);
20
```

The console output shows:

```
C:\Program Files\nodejs\node.exe .\codigo2.js
0
```

## Caso 2

- **Paso 2:** introduce un punto de interrupción para inspeccionar el valor de las variables '**i**' y '**factorial**'.  
Elabora una traza. Es decir, anota en cada iteración el valor de estas variables en una tabla con esta estructura:

Iteración	i	factorial
1	5	0
2	4	0
3	3	0
4	2	0
5	1	0
6	0	0

## Caso 2

- **Paso 3:** corrige el código para que se pueda realizar el factorial, en este caso de 5, y nos devuelva el resultado correcto. Explica en qué ha consistido la corrección e inspecciona de nuevo las variables para comprobar que es correcto.

```
let factorial=1;

for (i=limite;i>0;i--){
    factorial*=i;
}
```