4. Funciones de usuario y del sistema

4d. FUNCIONES DE USUARIO

1. Introducción

- Una función puede verse como una caja oscura a la que le pasamos una serie de valores, llamados parámetros, y devolverá uno o ningún valor como resultado de dicha función.

- A la hora de hacer un programa ligeramente grande existen determinados procesos que se pueden concebir de forma independiente, y que son más sencillos de resolver que el problema entero. Además, estos suelen ser realizados repetidas veces a lo largo de la ejecución del programa. Estos procesos se pueden agrupar en una función, definida para que no tengamos que repetir una y otra vez ese código en nuestros scripts, sino que simplemente llamamos a la función y ella se encarga de hacer todo lo que debe.

1. Introducción

- Las funciones se utilizan constantemente, no sólo las que escribes nosotros, sino también las que ya están definidas en el sistema, pues todos los lenguajes de programación tienen un montón de funciones para realizar procesos habituales como por ejemplo obtener la hora, imprimir un mensaje en la pantalla o convertir variables de un tipo a otro.

2. Método Clásico

- La instrucción **function** permite la definición de funciones. Después de esta palabra reservada se coloca el nombre de la función seguido de una lista de parámetros

```
function nombre (param1, param2,..., paramn){
   instrucciones en JavaScript;
}
```

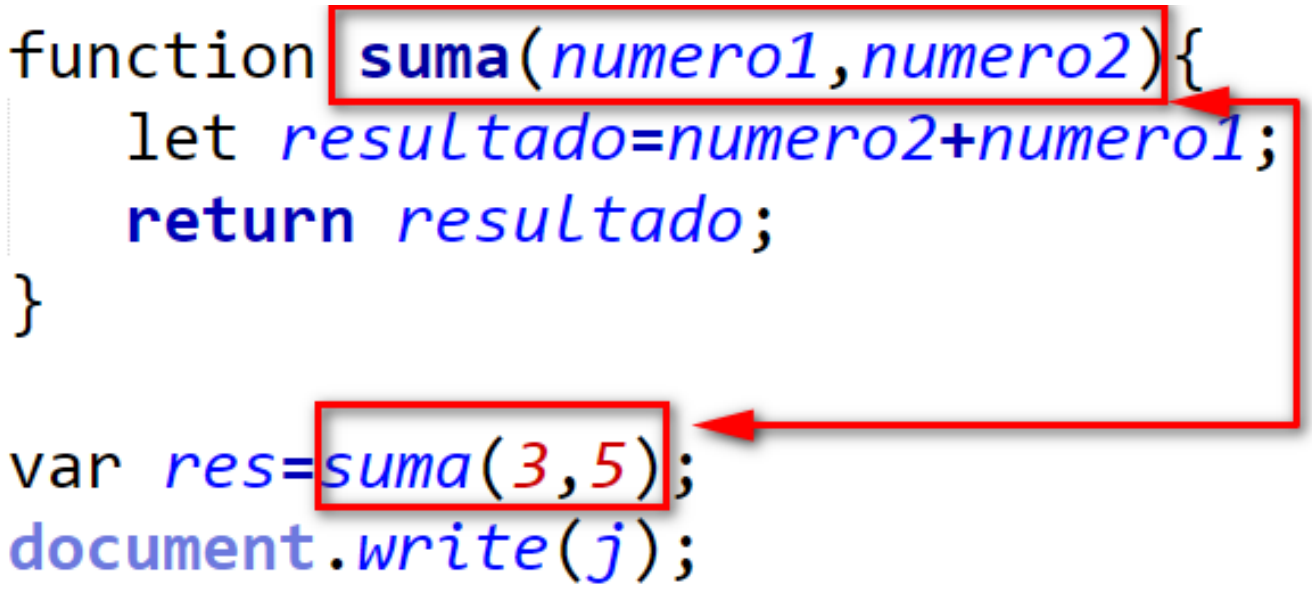
2. Método Clásico

- La sentencia **return** es la que permite devolver el resultado de una función. En el ejemplo que se verá a continuación, se muestra una función que devuelve el área de un cuadrado pasando como parámetro el lado.

```
function Area(Lado){
    return Lado*Lado;
}
```

2. Método Clásico

- Para llamar a la función debemos escribir su nombre, seguido de los parámetros que necesite dicha función.



3. Función anónima

- En la mayoría los tipos básicos (números y cadenas de caracteres principalmente) pueden integrarse en la lógica del código sin necesidad de ser almacenados en variables.

- Es decir, para sumar 2 y 3 podemos escribir el código 2 + 3 sin necesidad de guardar estos datos en variables. Pues bien, como las funciones son variables, **podemos crear una donde deseemos usarla sin necesidad de darle un nombre**. Esto es lo que se conoce como funciones anónimas.

3. Función anónima

- Pues bien, aprovechando esto podemos definir una variable como una función.
- En lugar de poner el nombre a la función, dicha función no tiene nombre y dicho nombre se asigna a una variable.

3. Función anónima

- Por ejemplo, tenemos la función suma:

```
function suma(a, b) {
    return a+b;
}
```

- Podemos convertirla a una función anónima asignada a una variable.

```
let suma=function (a,b) {
    return a+b;
}
```

3. Función anónima

- Por supuesto, el resultado es el mismo y la forma de llamarla, también.

```
suma(3,5);
```

4. Función Flecha

- Una variación de la forma anterior es reemplazar el termino **function** por una flecha **=>**
- Los parámetros irán en primer lugar (al no existir el termino función es lo primero que aparece) y tras los parámetros la citada flecha, el resto sería similar al caso anterior

4. Función Flecha

Con function	flecha

`let suma=function (a,b) {`
`  return a+b;`
`}`
	`let suma= (a,b) => {`
	`  return a+b;`
	`}`

- A efectos de ejecución, ambos ejemplos son iguales.