Las funciones

DigitalHouse >
Coding School

Índice

1 Declaración y estructura

Una función es un **bloque de código** que nos permite **agrupar funcionalidad** para usarla todas las veces que necesitemos.

Normalmente realiza una **tarea específica** y **retorna** un valor como resultado.

Estructura básica

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

Palabra reservada

Usamos la palabra **function** para indicarle a JavaScript que vamos a escribir una función.

Estructura básica

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

Nombre de la función

Definimos un nombre para referirnos a nuestra función al momento de querer **invocarla**.

Estructura básica

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

Parámetros

Escribimos los paréntesis y dentro de ellos los parámetros de la función. Si hay más de uno, los separamos usando comas `,`.

Si la función no lleva parámetros, igual debemos escribir los paréntesis sin nada adentro `()`.

Estructura básica

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

Parámetros

Dentro de nuestra función podremos acceder a los parámetros como si fueran variables. Es decir, con solo escribir los nombres de los parámetros, podremos trabajar con ellos.

Estructura básica

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

Cuerpo

Entre las llaves de apertura y de cierre escribimos la lógica de nuestra función, es decir, el código que queremos que se ejecute cada vez que la invoquemos.

Estructura básica

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

El retorno

Es muy común, a la hora de escribir una función, que queramos devolver al exterior el resultado del proceso que estamos ejecutando.

Para eso utilizamos la palabra reservada **return** seguida de lo que queramos retornar.

2 | Invocación

Podemos imaginar las funciones como si fueran máquinas.

Durante la **declaración** nos ocupamos de **construir** la máquina y durante la **invocación** la ponemos a **funcionar**.

Invocando una función

Antes de poder invocar una función, necesitamos que haya sido declarada. Vamos a declarar una función entonces...

```
{}  function hacerUnHelado() {
        return '🍦';
    }
```

La forma de **invocar** (también se puede decir llamar o ejecutar) una función es escribiendo su nombre seguido de apertura y cierre de paréntesis.

```
{}  hacerUnHelado(); // Retornará '🍦'
```

Invocando una función

Si la función tiene parámetros, se los podemos pasar dentro de los paréntesis cuando la invocamos. **Es importante respetar el orden** ya que JavaScript asignará los valores en el orden en que lleguen.

{}

```
function saludar(nombre, apellido) {
    return 'Hola ' + nombre + ' ' + apellido;
}

saludar('Robertito', 'Rodríguez');
// retornará 'Hola Robertito Rodríguez'
```

Invocando una función

También es importante tener en cuenta que cuando tenemos parámetros en nuestra función, JavaScript va a esperar que se los indiquemos al ejecutarla.

```
function saludar(nombre, apellido) {
    return 'Hola ' + nombre + ' ' + apellido;
}

saludar(); // retorna 'Hola undefined undefined'
```

En este caso, al no haber recibido el argumento que necesitaba, JavaScript le asigna el tipo de dato **undefined** a los parámetros *nombre* y *apellido*.

Invocando una función

Para casos como el anterior podemos definir **valores por defecto**.

Si agregamos un igual = luego un parámetro, podremos especificar su valor en caso de que no llegue ninguno.

```
{}
function saludar(nombre = 'visitante',
    apellido = 'anónimo') {
    return 'Hola ' + nombre + ' ' + apellido;
}

saludar(); // retornará 'Hola visitante anónimo'
```

Guardando los resultados

En caso de querer guardar lo que retorna una función, será necesario almacenarlo en una variable.

```
{}
function hacerHelados(cantidad) {
    return '🍦'.repeat(cantidad);
}

let misHelados = hacerHelados(3);
console.log(misHelados); // Mostrará en consola '🍦🍦🍦'
```

Llamamos **parámetros** a las **variables** que escribimos cuando **definimos** la función.

Llamamos **argumentos** a los **valores** que enviamos cuando **invocamos** la función.