

M1 - 7 - JS Funciones

Cuando se desarrolla una aplicación compleja, es muy habitual utilizar una y otra vez las mismas instrucciones. Un script para una tienda de comercio electrónico por ejemplo, tiene que calcular el precio total de los productos varias veces, para añadir los impuestos y los gastos de envío.

Cuando una serie de instrucciones se repiten una y otra vez, se complica demasiado el código fuente de la aplicación, ya que:

- El código de la aplicación es mucho más largo porque muchas instrucciones están repetidas.
- Si se quiere modificar alguna de las instrucciones repetidas, se deben hacer tantas modificaciones como veces se haya escrito esa instrucción, lo que se convierte en un trabajo muy pesado y muy propenso a cometer errores.

Las funciones son la solución a todos estos problemas, tanto en JavaScript como en el resto de lenguajes de programación. Una función es un conjunto de instrucciones que se agrupan para realizar una tarea concreta y que se pueden reutilizar fácilmente.

En el siguiente ejemplo, las instrucciones que suman los dos números y muestran un mensaje con el resultado se repiten una y otra vez:

```
let resultado;

let numero1 = 3;
let numero2 = 5;

// Se suman los números y se muestra el resultado
resultado = numero1 + numero2;
alert("El resultado es " + resultado);

numero1 = 10;
numero2 = 7;

// Se suman los números y se muestra el resultado
resultado = numero1 + numero2;
alert("El resultado es " + resultado);

numero1 = 5;
numero2 = 8;

// Se suman los números y se muestra el resultado
```

```
resultado = numero1 + numero2;
alert("El resultado es " + resultado);
```

Aunque es un ejemplo muy sencillo, parece evidente que repetir las mismas instrucciones a lo largo de todo el código no es algo recomendable. La solución que proponen las funciones consiste en extraer las instrucciones que se repiten y sustituirlas por una instrucción del tipo *"en este punto, se ejecutan las instrucciones que se han extraído"*.

Para que la solución del ejemplo anterior sea válida, las instrucciones comunes se tienen que agrupar en una función a la que se le puedan indicar los números que debe sumar antes de mostrar el mensaje.

Por lo tanto, en primer lugar se debe crear la función básica con las instrucciones comunes. Las funciones en JavaScript se definen mediante la palabra reservada **function**, seguida del nombre de la función. Su sintaxis es la siguiente:

```
function nombre_funcion() {
    ...
}
```

El nombre de la función se utiliza para *llamar* a esa función cuando sea necesario. El concepto es el mismo que con las variables, a las que se les asigna un nombre único para poder utilizarlas dentro del código. Después del nombre de la función, se incluyen dos paréntesis cuyo significado se detalla más adelante. Por último, los símbolos { y } se utilizan para encerrar todas las instrucciones que pertenecen a la función (de forma similar a como se encierran las instrucciones en las estructuras if o for).

Volviendo al ejemplo anterior, se crea una función llamada *suma_y_muestra* de la siguiente forma:

```
function suma_y_muestra() {
    resultado = numero1 + numero2;
    alert("El resultado es " + resultado);
}
```

Aunque la función está correctamente creada, no funciona como debería ya que le faltan los **"argumentos"**, que explicaremos a continuación. Una vez creada la función, desde cualquier punto del código se puede *llamar* a la función para que se ejecuten sus instrucciones.

La llamada a la función se realiza simplemente indicando su nombre, incluyendo los paréntesis del final y el carácter ; para terminar la instrucción:

Argumentos

Las funciones más sencillas no necesitan ninguna información para producir sus resultados. Sin embargo, la mayoría de funciones de las aplicaciones reales deben acceder al valor de algunas variables para producir sus resultados.

Las variables que necesitan las funciones se llaman **argumentos**. Antes de que pueda utilizarlos, la función debe indicar cuántos argumentos necesita y cuál es el nombre de cada argumento. Además, al invocar la función, se deben incluir los valores que se le van a pasar a la función. Los argumentos se indican dentro de los paréntesis que van detrás del nombre de la función y se separan con una coma (,).

Siguiendo el ejemplo anterior, la función debe indicar que necesita dos argumentos, correspondientes a los dos números que tiene que sumar:

```
function suma_y_muestra(primerNumero, segundoNumero) { ... }
```

A continuación, para utilizar el valor de los argumentos dentro de la función, se debe emplear el mismo nombre con el que se definieron los argumentos:

```
function suma_y_muestra(primerNumero, segundoNumero) { ... }  
    let resultado = primerNumero + segundoNumero;  
    alert("El resultado es " + resultado);  
}
```

Dentro de la función, el valor de la variable `primerNumero` será igual al primer valor que se le pase a la función y el valor de la variable `segundoNumero` será igual al segundo valor que se le pasa. Para pasar valores a la función, se incluyen dentro de los paréntesis utilizados al llamar a la función:

```
// Definición de la función  
function suma_y_muestra(primerNumero, segundoNumero) {  
    let resultado = primerNumero + segundoNumero;  
    alert("El resultado es " + resultado);  
}  
  
// Declaración de las variables  
let numero1 = 3;  
let numero2 = 5;  
  
// Llamada a la función  
suma_y_muestra(numero1, numero2);
```

En el código anterior, se debe tener en cuenta que:

- Aunque casi siempre se utilizan variables para pasar los datos a la función, se podría haber utilizado directamente el valor de esas variables: `suma_y_muestra(3, 5);`
- El número de argumentos que se pasa a una función debería ser el mismo que el número de argumentos que ha indicado al crear la función. No obstante, JavaScript **no muestra ningún error** si se pasan más o menos argumentos de los necesarios.
- El orden de los argumentos es fundamental, ya que el primer dato que se indica en la llamada, será el primer valor que espera la función; el segundo valor indicado en la llamada, es el segundo valor que espera la función y así sucesivamente.
- Se puede utilizar un número ilimitado de argumentos, aunque si su número es muy grande, se complica en exceso la llamada a la función.
- No es obligatorio que coincida el nombre de los argumentos que utiliza la función y el nombre de los argumentos que se le pasan. En el ejemplo anterior, los argumentos que se pasan son `numero1` y `numero2` y los argumentos que utiliza la función son `primerNumero` y `segundoNumero`.

A continuación se muestra otro ejemplo de una función que calcula el precio total de un producto a partir de su precio básico:

```
// Definición de la función
function calculaPrecioTotal(precio) {
  let impuestos = 1.16;
  let gastosEnvio = 10;
  var precioTotal = ( precio * impuestos ) + gastosEnvio;
}

// Llamada a la función
calculaPrecioTotal(23.34);
```

La función anterior toma como argumento una variable llamada `precio` y le suma los impuestos y los gastos de envío para obtener el precio total. Al llamar a la función, se pasa directamente el valor del precio básico mediante el número 23.34.

Afortunadamente, las funciones no solamente pueden recibir variables y datos, sino que también pueden devolver los valores que han calculado. Para devolver valores dentro de una función, se utiliza la palabra reservada **return**. Aunque las funciones pueden devolver valores de cualquier tipo, solamente pueden devolver un valor cada vez que se ejecutan.

```
function calculaPrecioTotal(precio) {
  let impuestos = 1.16;
  let gastosEnvio = 10;
  let precioTotal = ( precio * impuestos ) + gastosEnvio;
  return precioTotal;
}

let precioTotal = calculaPrecioTotal(23.34);
```

Para que la función devuelva un valor, solamente es necesario escribir la palabra reservada `return` junto con el nombre de la variable que se quiere devolver. En el ejemplo anterior, la ejecución de la función llega a la instrucción `return precioTotal;` y en ese momento, devuelve el valor que contenga la variable `precioTotal`.

Como la función devuelve un valor, en el punto en el que se realiza la llamada, debe indicarse el nombre de una variable en el que se guarda el valor devuelto.

Si no se indica el nombre de ninguna variable, JavaScript no muestra ningún error y el valor devuelto por la función simplemente se pierde y por tanto, no se utilizará en el resto del programa. En este caso, tampoco es obligatorio que el nombre de la variable devuelta por la función coincida con el nombre de la variable en la que se va a almacenar ese valor.

Si la función llega a una instrucción de tipo `return`, se devuelve el valor indicado y finaliza la ejecución de la función. Por tanto, todas las instrucciones que se incluyen después de un `return` se ignoran y por ese motivo la instrucción `return` suele ser la última de la mayoría de funciones.

Para que el ejemplo anterior sea más completo, se puede añadir otro argumento a la función que indique el porcentaje de impuestos que se debe añadir al precio del producto.

Evidentemente, el nuevo argumento se debe añadir tanto a la definición de la función como a su llamada:

```
function calculaPrecioTotal(precio, porcentajeImpuestos) {
  let gastosEnvio = 10;
  let precioConImpuestos = (1 + porcentajeImpuestos/100) * precio;
  let precioTotal = precioConImpuestos + gastosEnvio;
  return precioTotal;
}

let precioTotal = calculaPrecioTotal(23.34, 16);
let otroPrecioTotal = calculaPrecioTotal(15.20, 4);
```