14 Funciones

Las funciones son la base de la modularidad en JavaScript. Son utilizadas para reutilizar código, ocultar información y abstracción. Por norma general, las funciones son utilizadas para especificar el comportamiento de los objetos, aunque pueden definirse funciones al margen de los objetos.

Su sintaxis básica de declaración es:

```
function identificador (argumento1, argumento2, ...) {
        // código a ejecutar por la función (conocido también como cuerpo)
}
```

El código que realizan las funciones no es ejecutado hasta que las funciones no son invocadas (llamadas), proceso que se realiza cuando utilizamos el identificador de la función seguido de paréntesis y argumentos en el caso de que los tenga.

```
identificador(argumento1, argumento2);
```

Las funciones pueden devolver un valor o no, en el primer caso, la última instrucción del cuerpo de la función será la palabra reservada return más la expresión o valor que queramos que devuelva.

```
function identificador (argumento1, argumento2, ...) {
        // código a ejecutar por la función
        return expresión
}
```

Funciones anónimas

```
let variable = function (argumento1, argumento2, ...) {
        // código a ejecutar por la función
        return expresión
}

variable(argumentos);
```

Callbacks

Un callback (llamada de vuelta) es una función que recibe como argumento otra función y la ejecuta. Los callbacks son funciones que son invocadas cuando desde otra función ha terminado de ejecutarse alguna tarea específica.

Ámbito de las variables y constantes

El ámbito de una variable (llamado "scope" en inglés) es la zona del programa en la que se define la variable. JavaScript define dos ámbitos para las variables: global y local.

A la hora de decidir a qué ámbito pertenece una variable, el intérprete de JavaScript comprueba si la variable está declarada dentro de la función donde se invoca (ámbito local de la función) si no es así, busca en el ámbito de la función que la contenga y así hasta llegar al ámbito global. Si la variable no fue declarada en ningún momento, el compilador la declara como variable global.

En ES6 se introdujeron los tipos de variable let y const que no funcionan como las variables var sino que tienen ámbito de bloque (block scope). Esto significa que estas variables existen dentro del bloque donde son declaradas, independientemente de si ese bloque es una función, una condición o un bucle. Además, las variables declaradas con let y const no se pueden utilizar antes de ser declaradas.