Capítulo 5
Funciones

1. Introducción

Una función le permite escribir varias instrucciones para que pueda reutilizar el bloque de instrucciones sin volver a escribir cada una de ellas. Se nombra y define una vez, pero se ejecuta varias veces en el programa principal.

Las funciones ya se han utilizado en este libro. El curso normal del programa es una función: la función principal. `document.writeln()` es una función, que sirve para mostrar algo en la página web.

2. Funciones simples

2.1 Sintaxis

Una función se declara usando la palabra clave `function` seguida de su nombre.

```
function miFuncion() {
        operacion(es);
}
```

A cada llamada, `miFuncion()` realizará las operaciones que incluya. Para llamar a una función, simplemente escriba su nombre.

```
<!-- C:\JavaScriptYAngular\JavaScript\C05\demostracion\index.html -->
<!DOCTYPE html>
<html lang="es">

<head>
   <meta charset="UTF-8">
   <title>Ediciones ENI</title>
</head>

<body>
   <script type="text/javascript">
       // Definición de la función
       function fLeyes() {
           document.writeln("Un robot no puede dañar a un ser humano, ni
permanecer pasivo mientras un ser humano está expuesto al peligro. <br />")
           document.writeln("Un robot debe obedecer las órdenes que le son
dadas por un ser humano, a menos que tales órdenes entren en conflicto con
la primera ley<br />");
          document.writeln("Un robot debe proteger su existencia mientras
esta protección no entre en conflicto con la primera o segunda ley.<br />");
           document.writeln("--- Isaac ASIMOV - Círculo vicioso ---
<br />");
           document.writeln("<br /><br />");
       }
       // Llamada simple
       document.writeln("<h3>Llamada a una función</h3>")
       fLeyes ();
       // Llamada dentro de una función
       document.writeln("<h3> Llamar a una función en un bucle for</h3>")
       for (let i = 1; i < 10; i++) {
           document.writeln("<h4>" + i + "</h4>")
           fLeyes ();
       }
   </script>
</body>

</html>
```

2.2 Ámbito

2.2.1 Ámbito global

Una función puede acceder a una variable definida fuera de esta función.

```
<!-- C:\JavaScriptYAngular\JavaScript\C05\demostracion\
index.html -->
<!DOCTYPE html>
<html lang="es">

<head>
   <meta charset="UTF-8">
   <title> Ediciones ENI</title>
</head>

<body>
   <script type="text/javascript">
       // Funciones :
       // fMultiplicacion - Multiplicación de un número fuera
de la función por 1337
       function fMultiplicacion() {
           leet = 1337;
           document.writeln(numeroDeGeek + "*" + leet + "=" +
numeroDeGeek * leet);
       }

       // Llamadas
       // Llamada de fMultiplicacion
       document.writeln("<h4>fMultiplicacion</h4>");
       numeroDeGeek = 56;
       fMultiplicacion();
   </script>
</body>

</html>
```

Observación

56 es el número de geeks identificados por Scott Johnson en un famoso cartel llamado The 56 geeks project.

La variable `numeroDeGeek` definida en el programa principal es de alcance global. Por lo tanto, está disponible absolutamente en todas partes del programa, incluidas las funciones.

2.2.2 Ámbito local

Una variable con alcance local se define dentro de una función. Solo es accesible dentro de él.

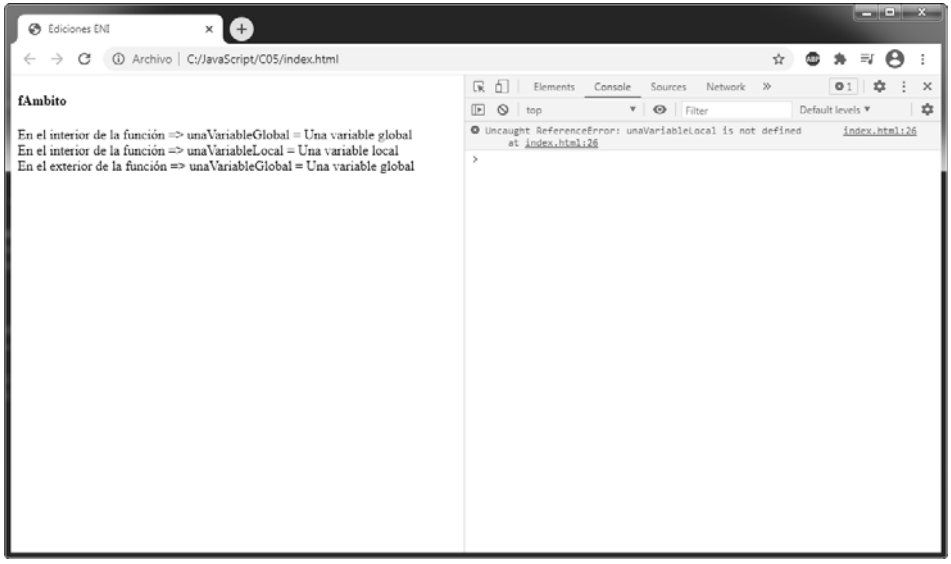
```
<!-- C:\JavaScriptYAngular\JavaScript\C05\demostracion\
index.html -->
<!DOCTYPE html>
<html lang="es">

<head>
   <meta charset="UTF-8">
   <title> Ediciones ENI</title>
</head>

<body>
   <script type="text/javascript">
       // Funciones:
       // fAmbito - Prueba de ambito de una variable local
       function fAmbito() {
           let unaVariableLocal = "Una variable local";
           document.writeln("En el interior de la función =>
unaVariableGlobal = " + unaVariableGlobal + "<br />");
           document.writeln("En el interior de la función =>
unaVariableLocal = " + unaVariableLocal + "<br />");
       }
       // Llamadas
       // Llamada de fAmbito
       document.writeln("<h4>fAmbito</h4>");
       let unaVariableGlobal = "Una variable global";
       fAmbito();
       document.writeln("En el exterior de la función =>
unaVariableGlobal = " + unaVariableGlobal + "<br />");
       document.writeln("En el exterior de la función =>
unaVariableLocal = " + unaVariableLocal + "<br />");
   </script>
</body>

</html>
```

Dentro de la función, las variables globales y locales se interpretan y muestran bien. Por otro lado, el procesamiento de la variable local fuera de la función provoca un error en la consola del navegador.



Demostración del ámbito de una variable local

3. Funciones y argumentos

Al llamar a una función, es posible definir uno o más argumentos utilizados como variables dentro de ella. Estas variables tienen un alcance limitado al bloque de instrucciones de la función. Los argumentos son una forma de pasar información o datos de la función principal a una función secundaria. Una función puede acomodar un número ilimitado de argumentos de cualquier tipo.

3.1 Sintaxis de la función con argumentos

Los argumentos se declaran entre paréntesis ubicados después del nombre de la función y no se escriben. Como recordatorio, JavaScript no es un lenguaje de programación fuertemente tipado y es el uso que hacemos de las variables lo que define sus tipos.

```
function funcionesYArgumentos(argumento01, argumento02, argumento03) {
     console.log(argumento01);
     console.log(argumento02);
     console.log(argumento03);
}
```

3.2 Llamada a la función con argumentos

Por lo tanto, para llamar a una función con argumentos, simplemente escriba su nombre seguido de los argumentos.

```
<!-- C:\JavaScriptYAngular\JavaScript\C05\demostracion\
index.html -->
<!DOCTYPE html>
<html lang="es">

<head>
   <meta charset="UTF-8">
   <title> Ediciones ENI</title>
</head>

<body>
   <script type="text/javascript">
       // Funciones :
       // fArgumentos
       function fArgumentos(argumento01, argumento02, argumento03) {
          document.writeln("argumento01 => " + argumento01 + " <br/>");
          document.writeln("argumento02 => " + argumento02 + " <br/>");
          document.writeln("argumento03 => " + argumento03 + " <br/>");
       }
       // Llamadas
       // Llamada de fArgumentos
       document.writeln("<h3>Funciones con argumentos</h3>");
       let unaCadema = "Una cadena de caracteres";
       let unNumero = 1337;
       let unArray = ["Rafael", "Anaís", "Rosa", "Ana-Martina"];
```

```
        fArgumentos(unaCadena, unNumero, unArray);
    </script>
</body>

</html>
```

3.3 Valor por defecto de los argumentos

De forma predeterminada, un argumento es de tipo `indefinido`. Este comportamiento puede generar un error al utilizar la función. Por ejemplo, si el desarrollador, que llama a la función, proporciona un número de argumentos menor que el número requerido y el cuerpo de la función realiza operaciones, entonces es casi inevitable que aparezca un error.

Imaginemos una función que realiza una división entre dos argumentos.

```
function fDivision(argumento01, argumento02) {
           document.writeln("La división del primer argumento por
el segundo= " + argumento01 / argumento02);
}
```

Si el desarrollador llama a la función de esta manera:

```
fDivision(50);
```

entonces `argumento02` es indefinido y dividir 50 por algo indefinido no devuelve un número.

Para resolver este tipo de problema, es común ver un valor predeterminado de los argumentos en la firma de la función. En este ejemplo, podemos suponer que el desarrollador está distraído o realmente no quiere dividir un número y, por lo tanto, el valor predeterminado del divisor es 1.

Para definir el valor predeterminado de un argumento, todo lo que tiene que hacer es ingresarlo en la firma de la función junto a su nombre, con el operador `=`.

```
<!-- C:\JavaScriptYAngular\JavaScript\C05\demostracion\
index.html -->
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">
```