FUNCIONES

Las funciones en JavaScript, así como en otros lenguajes de programación no son más que porciones de código independientes que realizan una tarea específica.

La división en la etapa de diseño de un programa en subrutinas o subprogramas y su posterior implementación utilizando funciones bien determinadas, facilitan la labor del programador a la hora de corregir errores tanto de sintaxis como lógicos.

El desarrollo, la escalabilidad y el mantenimiento de un programa "modular" como suele llamársele a un programa implementado con funciones, también se facilita.

Una vez definida y comprobado el correcto desempeño de una función, ésta puede ser utilizada tantas veces como sea necesario y desde cualquier parte del programa.

Una función en JavaScript, debe ser definida para luego ser llamada desde cualquier parte del programa. De nuevo, se recomienda hacer la definición de todas las funciones en la sección HEAD del documento HTML, para evitar errores generados por el llamado a funciones que aún no han sido interpretadas.

Las funciones en JavaScript pueden o no devolver valores, así como, pueden o no recibir argumentos para realizar sus tareas. La siguiente es la forma general de declarar o definir una función en JavaScript:

```
function nombreFuncion (argumento1, argumento2, ... argumentoN)
{
    sentencia1;
    sentencia2;
    ...
    sentenciaN;
return expresión; //Si la función devuelve un valor
}
```

Los nombres de las funciones en JavaScript se asignan teniendo en cuenta las mismas reglas descritas para nombrar variables.

Con el siguiente script, se implementa una función llamada escribeDatos, que no recibe argumentos ni devuelve ningún valor, y cuya tarea consiste en imprimir en el cuerpo de la página el nombre del usuario y la edad del mismo, basada en la fecha de nacimiento y el año actual:

```
<script language="JavaScript1.2">

    function escribeDatos()
    {
          var nombre="Andrés Gaitán";
          var nacimiento=1986;
          var actual=2004;
          document.write("Hola, soy " + nombre + " y tengo " +
                         (actual-nacimiento) + " años");
    }

</script>
```

Una vez definida, una función se puede utilizar tantas veces como sea necesario. Para que una función ejecute la tarea para la cual fue definida, es indispensable hacer un llamado a la misma. Los llamados a funciones como en todos los lenguajes, se realizan utilizando el nombre de la función y la lista de argumentos entre paréntesis:

```
nombre_funcion(argumento1, argumento2, ... argumentoN);
```

En el ejemplo, la función no recibe ningún argumento, por lo que el llamado a escribeDatos se hace de la siguiente manera:

```
escribeDatos();
```

El listado siguiente contiene el código necesario para elaborar una página que calcula y muestra la edad de Andrés:

```
<HTML>
<HEAD>
<TITLE>Datos</TITLE>
<SCRIPT language="JavaScript1.2">
    //Definición de la función escribeDatos
    function escribeDatos()
    {
      var nombre="Andrés Gaitán";
      var nacimiento=1986;
      var actual=2004;
      document.write("Hola, soy " + nombre + " y tengo " +
                     (actual-nacimiento) + " años");
    }
</SCRIPT>
</HEAD>

<BODY>

    <SCRIPT language="JavaScript1.2">
          //Llamado a la función escribeDatos
          escribeDatos();
    </SCRIPT>

</BODY>
</HTML>
```

Para ilustrar la manera de definir y llamar funciones que reciben argumentos, en el siguiente listado se han hecho las modificaciones necesarias para que la función escribeDatos del ejemplo anterior, reciba en el llamado los valores correspondientes a los argumentos nombre, actual y nacimiento:

```
<HTML>
<HEAD>
<TITLE>Datos</TITLE>
<SCRIPT language="JavaScript1.2">
    //Definición de la función escribeDatos. Recibe tres argumentos
    function escribeDatos(nombre, actual, nacimiento)
    {
       document.write("Hola, soy " + nombre + " y tengo " +
                      (actual-nacimiento) + " años");
    }
</SCRIPT>
</HEAD>

<BODY>
    <SCRIPT language="JavaScript1.2">
          //Llamado a la función escribeDatos con los 3 argumentos
          escribeDatos("Andrés Gaitán", 2004, 1986);
    </SCRIPT>
</BODY>
</HTML>
```

Dentro del bloque de sentencias de una función puede existir cualquier expresión válida, incluso el llamado a otra función previamente definida. En el siguiente ejemplo se definen dos funciones: la función calcula, que recibe como argumentos el año actual y el año de nacimiento y devuelve la diferencia entre ambos valores. La función escribeDatos que recibe como argumento el nombre del usuario y lo imprime junto con la edad calculada por la función calcula:

```
<HTML>
<HEAD>
<TITLE>Datos</TITLE>
<SCRIPT language="JavaScript1.2">

    //Definición de la función calcula
    function calcula(actual, nacimiento)
    {
    //La función devuelve la diferencia entre actual y nacimiento
          return (actual-nacimiento);
    }

    //Definición de la función escribeDatos
    function escribeDatos(nombre)
    {
       /*La variable edad recibe el resultado devuelto por calcula,
         que es llamada con los argumentos actual y nacimiento */
       edad = calcula(2004, 1986);
       document.write("Hola, soy " + nombre + " y tengo " +
                      edad + " años");
    }
</SCRIPT>
</HEAD>

<BODY>

    <SCRIPT language="JavaScript1.2">
    //Llamado a la función escribeDatos con el argumento nombre
          escribeDatos("Andrés Gaitán");
    </SCRIPT>

</BODY>
</HTML>
```

Variables globales y variables locales

Existen diferencias en el comportamiento de una variable dependiendo del sitio donde ha sido declarada. Cuando la declaración se hace fuera de cualquier función, se dice que la variables es global y puede ser utilizada desde cualquier función definida dentro del mismo script. Mientras que si la variable se declara dentro de una función específica, la variable es local y el uso de la misma está limitado a operaciones dentro del ámbito de la función donde fue declarada. Si se intenta utilizar una variable local por fuera de la función donde fue declarada, el resultado será un error en tiempo de ejecución.

```
<HTML>
<HEAD>
<TITLE>Variables</TITLE>
    <SCRIPT language="JavaScript1.2">
      //Declaración de la variable global x
      var x=5;
      //Definición de la función suma
      function suma()
      {
          //Declaración de la variable local y
          var y=5;
          r=x+y;
          document.write("X+Y="+r);
      }

      function resta()
      {
          //La siguiente línea generará error, la variable y es
            indefinida
          r=x-y;
          //Lo siguiente no se imprimirá
          document.write("X-Y="+(x-y));
      }

      suma();
      resta();
    </SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Funciones predefinidas en JavaScript

Existen algunas funciones predefinidas en JavaScript, la mayoría de las cuales permiten chequear el tipo de dato de una expresión o variable que se pasa como argumento a la función, evaluar expresiones o hacer casting de tipos de datos:

La ***función eval*** permite la evaluación de cadenas de caracteres. Si la cadena de caracteres representa una expresión, la función eval devuelve el resultado de evaluar la expresión. Por ejemplo:

```
//Y es una cadena de caracteres
Y="35+56/28+1.5";
//Si imprime el valor de Y
document.write(Y);
//El resultado será 35+56/28+1.5

//Ahora X recibe como valor el resultado de evaluar Y
X=eval(Y);

//Si imprime X
document.write(X);
//El resultado será 38.5
```

Si la cadena de caracteres contiene sentencias válidas de JavaScript , el resultado de la evaluación será la ejecución de las sentencias contenidas en la cadena. Veamos:

```
//La variable Y vale 5;
Y=5;
//Z es una cadena de caracteres que contiene 2 sentencias
Z="Y+=10; document.write(Y);"
//La función eval ejecuta las dos sentencias contenidas en Z
eval(Z);
//El resultado es la impresión del número 15
```

La ***función isFinite()*** evalúa la expresión que se le pasa como argumento y devuelve un valor booleano dependiendo del resultado de la evalución. Devuelve `false` cuando la expresión no es un valor numérico o cuando el resultado es un valor infinito, por ejemplo, una división por cero. Devuelve `true` en cualquier otro caso:

```
x=5;
y="Hola";
z=0;
//La siguiente línea imprimirá false
document.write(isFinite(y+x));
//La siguiente línea imprimirá false
document.write(isFinite(x/z));
//La siguiente línea imprimirá true
document.write(isFinite(x*z));
```

La función ***isNaN()*** devuelve `true` cuando la expresión que se le pasa como argumento no es un valor numérico, y `false` en cualquier otro caso:

```
x=5;
y="Hola";
z=0;
//La siguiente línea imprimirá true
document.write(isNaN(y+x));
//La siguiente línea imprimirá false
document.write(isNaN(x/z));
//La siguiente línea imprimirá false
document.write(isNaN(x*z));
```

Las función ***parseInt()*** devuelve un entero correspondiente al valor de la cadena de caracteres que se le pasa como argumento, si esta puede ser evaluada, de lo contrario devuelve NaN. Además, esta función permite convertir números entre diferentes bases:

```
X="50.5"
//Z toma como valor 50
Z=parseInt(X) ;
/*Z ahora vale 40 en base 10que es el resultado de convertir 50 que
está expresado en base 8 */
Z=parseInt(X, 8);
/*Z ahora vale 80 en base 10 que es el resultado de convertir 50 que
está expresado en base 16 */
Z=parseInt(X, 16);
```

La función ***parseFloat()*** trabaja de manera similar a *parseInt()*. Devuelve un valor real, producto de la evaluación de la cadena que se le pasa como argumento. Ambas funciones devuelven `NaN`, si el primer carácter de la cadena es diferente a un dígito, un punto (.), el signo mas (+) o el signo menos (-). Si los primeros caracteres corresponden a una expresión numérica válida y los siguientes son otros caracteres cualesquiera, estos serán ignorados. Por ejemplo, en la expresión `z=parseFloat("-3.58e-3 Hola"); z` tomará como valor `-0.0038`, que corresponde a -3.58×10^{-3}, los caracteres `" Hola"` son ignorados por la función.

Existen otras cuatro funciones predefinidas en JavaScript, que serán explicadas o utilizadas más adelante, lo mismo que las funciones aritméticas que fueron implementadas en JavaScript como objetos.