Introducción a JavaScript

MANUAL II. FUNCIONES

Funciones

En programación es muy frecuente que un determinado procedimiento de cálculo definido por un grupo de sentencias tenga que repetirse varias veces, ya sea en un mismo programa o en otros programas, lo cual implica que se tenga que escribir tantos grupos de aquellas sentencias como veces aparezca dicho proceso.
La herramienta más potente con que se cuenta para facilitar, reducir y dividir el trabajo en programación, es escribir aquellos grupos de sentencias una sola y única vez bajo la forma de una FUNCION.

Un programa es una cosa compleja de realizar y por lo tanto es importante que esté bien ESTRUCTURADO y también que sea inteligible para las personas. Si un grupo de sentencias realiza una tarea bien definida, entonces puede estar justificado el aislar estas sentencias formando una función, aunque resulte que sólo se le llame o use una vez.
Hasta ahora hemos visto como resolver un problema planteando un único algoritmo. Con funciones podemos segmentar un programa en varias partes. Frente a un problema, planteamos un algoritmo, éste puede constar de pequeños algoritmos.

Una función es un conjunto de instrucciones que resuelven una parte del problema y que puede ser utilizado (llamado) desde diferentes partes de un programa. Consta de un nombre y parámetros. Con el nombre llamamos a la función, es decir, hacemos referencia a la misma. Los parámetros son valores que se envían y son indispensables para la resolución del mismo. La función realizará alguna operación con los parámetros que le enviamos. Podemos cargar una variable, consultarla, modificarla, imprimirla, etc.

Incluso los programas más sencillos tienen la necesidad de fragmentarse. Las funciones son los únicos tipos de subprogramas que acepta JavaScript. Tienen la siguiente estructura:

```
function <nombre de función>(argumento1, argumento2, ..., argumento n)
{
  <código de la función>
}
```

Debemos buscar un nombre de función que nos indique cuál es su objetivo (Si la función recibe un string y lo centra, tal vez deberíamos llamarla centrarTitulo). Veremos que una función puede variar bastante en su estructura, puede tener o no parámetros, retornar un valor, etc.

Ejemplo: Mostrar un mensaje que se repita 3 veces en la página con el siguiente texto:

```
'Cuidado'
'Ingrese su documento correctamente'

'Cuidado'
'Ingrese su documento correctamente'

'Cuidado'
'Ingrese su documento correctamente'
```

La solución sin emplear funciones es:

```
<html>
<head>
</head>
<body>

<script type="text/javascript">
  document.write("Cuidado<br>");
  document.write("Ingrese su documento correctamente<br>");
  document.write("Cuidado<br>");
  document.write("Ingrese su documento correctamente<br>");
  document.write("Cuidado<br>");
  document.write("Ingrese su documento correctamente<br>");
</script>

</body>
</html>
```

Empleando una función:

```
<html>
<head>
</head>
<body>

<script type="text/javascript">
  function mostrarMensaje()
  {
    document.write("Cuidado<br>");
    document.write("Ingrese su documento correctamente<br>");
  }

  mostrarMensaje();
  mostrarMensaje();
  mostrarMensaje();
</script>

</body>
</html>
```

Recordemos que JavaScript es sensible a mayúsculas y minúsculas. Si fijamos como nombre a la función mostrarTitulo (es decir la segunda palabra con mayúscula) debemos respetar este nombre cuando la llamemos a dicha función.
Es importante notar que para que una función se ejecute debemos llamarla desde fuera por su nombre (en este ejemplo: mostrarMensaje()).

Cada vez que se llama una función se ejecutan todas las líneas contenidas en la misma. Si no se llama a la función, las instrucciones de la misma nunca se ejecutarán. A una función la podemos llamar tantas veces como necesitemos.

Las funciones nos ahorran escribir código que se repite con frecuencia y permite que nuestro programa sea más entendible.

Funciones con parámetros.

Explicaremos con un ejemplo, una función que tiene datos de entrada.
Ejemplo: Confeccionar una función que reciba dos números y muestre en la página los valores comprendidos entre ellos de uno en uno. Cargar por teclado esos dos valores.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">

  function mostrarComprendidos(x1,x2)
  {
    var inicio;
    for(inicio=x1;inicio<=x2;inicio++)
    {
      document.write(inicio+' ');
    }
  }

  var valor1,valor2;
  valor1=prompt('Ingrese valor inferior:','');
  valor1=parseInt(valor1);
  valor2=prompt('Ingrese valor superior:','');
  valor2=parseInt(valor2);
  mostrarComprendidos(valor1,valor2);


</script>
</body>
</html>
```

El programa de JavaScript empieza a ejecutarse donde definimos las variables valor1 y valor2 y no donde se define la función. Luego de cargar los dos valores por teclado se llama a la función mostrarComprendidos y le enviamos las variables valor1 y valor2. Los parámetros x1 y x2 reciben los contenidos de las variables valor1 y valor2.
Es importante notar que a la función la podemos llamar la cantidad de veces que la necesitemos.

Los nombres de los parámetros, en este caso se llaman x1 y x2, no necesariamente se deben llamar igual que las variables que le pasamos cuando la llamamos a la función, en este caso le pasamos los valores valor1 y valor2.

Funciones que retornan un valor.

Son comunes los casos donde una función, luego de hacer un proceso, retorne un valor. Ejemplo 1: Confeccionar una función que reciba un valor entero comprendido entre 1 y 5. Luego retornar en castellano el valor recibido.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">

  function convertirCastellano(x)
  {
    if (x==1)
      return "uno";
    else
      if (x==2)
        return "dos";
      else
        if (x==3)
          return "tres";
        else
          if (x==4)
            return "cuatro";
          else
            if (x==5)
              return "cinco";
            else
              return "valor incorrecto";
  }

  var valor;
  valor=prompt("Ingrese un valor entre 1 y 5","");
  valor=parseInt(valor);
  var r;
  r=convertirCastellano(valor);
  document.write(r);

</script>
</body>
</html>
```

Podemos ver que el valor retornado por una función lo indicamos por medio de la palabra clave return. Cuando se llama a la función, debemos asignar el nombre de la función a una variable, ya que la misma retorna un valor.
Una función puede tener varios parámetros, pero sólo puede retornar un único valor. La estructura condicional if de este ejemplo puede ser remplazada por la instrucción switch, la función queda codificada de la siguiente manera:

```
function convertirCastellano(x)
{
  switch (x)
  {
    case 1:return "uno";
    case 2:return "dos";
    case 3:return "tres";
    case 4:return "cuatro";
    case 5:return "cinco";
    default:return "valor incorrecto";
  }
}
```

Esta es una forma más elegante que una serie de if anidados. La instrucción switch analiza el contenido de la variable x con respecto al valor de cada caso. En la situación de ser igual, ejecuta el bloque seguido de los 2 puntos hasta que encuentra la instrucción return o break.

Ejemplo 2: Confeccionar una función que reciba una fecha con el formato de día, mes y año y retorne un string con un formato similar a: "Hoy es 10 de junio de 2013".

```
<html>
<head>
</head>
<body>
<script type="text/javascript">

  function formatearFecha(dia,mes,año)
  {
    var s='Hoy es '+dia+' de ';
    switch (mes) {
    case 1:s=s+'enero ';
           break;
    case 2:s=s+'febrero ';
           break;
    case 3:s=s+'marzo ';
           break;
    case 4:s=s+'abril ';
           break;
    case 5:s=s+'mayo ';
           break;
    case 6:s=s+'junio ';
           break;
    case 7:s=s+'julio ';
           break;
    case 8:s=s+'agosto ';
           break;
    case 9:s=s+'septiembre ';
           break;
    case 10:s=s+'octubre ';
           break;
    case 11:s=s+'noviembre ';
           break;
    case 12:s=s+'diciembre ';
           break;
    } //fin del switch
```

```
    s=s+'de '+año;
    return s;
  }

  document.write(formatearFecha(11,6,2013));

</script>
</body>
</html>
```

Analicemos un poco la función formatearFecha. Llegan tres parámetros con el día, mes y año. Definimos e inicializamos una variable con:

```
    var s='Hoy es '+dia+' de ';
```

Luego le concatenamos o sumamos el mes:

```
s=s+'enero ';
```

Esto, si el parámetro mes tiene un uno. Observemos como acumulamos lo que tiene 's' más el string 'enero '. En caso de hacer s='enero ' perderíamos el valor previo que tenía la variable s.

Por último concatenamos el año:

```
s=s+'de '+año;
```

Cuando se llama a la función directamente, al valor devuelto se lo enviamos a la función write del objeto document. Esto último lo podemos hacer en dos pasos:

```
  var fec= formatearFecha(11,6,2013);
  document.write(fec);
```

Guardamos en la variable 'fec' el string devuelto por la función.