Valores de retorno en funciones Javascript

Las funciones pueden devolver valores, a través de la sentencia return. También vemos un apunte sobre el ámbito de variables en funciones en Javascript.

Estamos aprendiendo acerca del uso de funciones en Javascript y en estos momentos quizás ya nos hayamos dado cuenta de la gran importancia que tienen para hacer programas más o menos avanzados. En este artículo del Manual de Javascript seguiremos aprendiendo cosas sobre funciones y en concreto que con ellas también se puede devolver valores. Además, veremos algún caso de uso interesante sobre las funciones que nos puede aclarar un poco el ámbito de variables locales y globales.

Devolución de valores en las funciones

Las funciones en Javascript también pueden retornar valores. De hecho, ésta es una de las utilidades más esenciales de las funciones, que debemos conocer, no sólo en Javascript sino en general en cualquier lenguaje de programación. De modo que, al invocar una función, se podrá realizar acciones y ofrecer un valor como salida.

Por ejemplo, una función que calcula el cuadrado de un número tendrá como entrada a ese número y como salida tendrá el valor resultante de hallar el cuadrado de ese número. La entrada de datos en las funciones la vimos anteriormente en el artículo sobre parámetros de las funciones. Ahora tenemos que aprender acerca de la salida.

Veamos un ejemplo de función que calcula la media de dos números. La función recibirá los dos números y retornará el valor de la media.

```
function media(valor1,valor2){
        var resultado
        resultado = (valor1 + valor2) / 2
        return resultado
}
```

Para especificar el valor que retornará la función se utiliza la palabra return seguida de el valor que se desea devolver. En este caso se devuelve el contenido de la variable resultado, que contiene la media calculada de los dos números.

Quizás nos preguntemos ahora cómo recibir un dato que devuelve una función. Realmente en el código fuente de nuestros programas podemos invocar a las funciones en el lugar que

deseemos. Cuando una función devuelve un valor simplemente se sustituye la llamada a la función por ese valor que devuelve. Así pues, para almacenar un valor de devolución de una función, tenemos que asignar la llamada a esa función como contenido en una variable, y eso lo haríamos con el operador de asignación =.

Para ilustrar esto se puede ver este ejemplo, que llamará a la función media() y guardará el resultado de la media en una variable para luego imprimirla en la página.

```
var miMedia
miMedia = media(12,8)
document.write (miMedia)
```

Múltiples return
En realidad en Javascript las funciones sólo pueden devolver un valor, por lo que en principio no podemos hacer funciones que devuelvan dos datos distintos.

Nota: en la práctica nada nos impide que una función devuelva más de un valor, pero como sólo podemos devolver una cosa, tendríamos que meter todos los valores que queremos devolver en una estructura de datos, como por ejemplo un array. No obstante, eso sería un uso más o menos avanzado que no vamos a ver en estos momentos.

Ahora bien, aunque sólo podamos devolver un dato, en una misma función podemos colocar más de un return. Como decimos, sólo vamos a poder retornar una cosa, pero dependiendo de lo que haya sucedido en la función podrá ser de un tipo u otro, con unos datos u otros.

En esta función podemos ver un ejemplo de utilización de múltiples return. Se trata de una función que devuelve un 0 si el parámetro recibido era par y el valor del parámetro si este era impar.

```
function multipleReturn(numero){
        var resto = numero % 2
        if (resto == 0)
        return 0
        else
        return numero }
```

Para averiguar si un número es par hallamos el resto de la división al dividirlo entre 2. Si el resto es cero es que era par y devolvemos un 0, en caso contrario -el número es impar- devolvemos el parámetro recibido.

Ámbito de las variables en funciones
Dentro de las funciones podemos declarar variables. Sobre este asunto debemos de saber que todas las variables declaradas en una función son locales a esa función, es decir, sólo tendrán validez durante la ejecución de la función.

Nota: Incluso, si lo pensamos, nos podremos dar cuenta que los parámetros son como variables que se declaran en la cabecera de la función y que se inicializan al llamar a la función. Los parámetros también son locales a la función y tendrán validez sólo cuando ésta se está ejecutando.
Podría darse el caso de que podemos declarar variables en funciones que tengan el mismo nombre que una variable global a la página. Entonces, dentro de la función, la variable que tendrá validez es la variable local y fuera de la función tendrá validez la variable global a la página.

En cambio, si no declaramos las variables en las funciones se entenderá por javascript que estamos haciendo referencia a una variable global a la página, de modo que si no está creada la variable la crea, pero siempre global a la página en lugar de local a la función.

Veamos el siguiente código.

```
function variables_glogales_y_locales(){
   var variableLocal = 23
   variableGlobal = "qwerty"
}
```

En este caso variableLocal es una variable que se ha declarado en la función, por lo que será local a la función y sólo tendrá validez durante su ejecución. Por otra parte variableGlobal no se ha llegado a declarar (porque antes de usarla no se ha utilizado la palabra var para declararla). En este caso la variable variableGlobal es global a toda la página y seguirá existiendo aunque la función finalice su ejecución. Además, si antes de llamar a la función existiese la variable variableGlobal, como resultado de la ejecución de esta función, se machacaría un hipotético valor de esa variable y se sustituiría por "qwerty".