Programación con Javascript

Unidad 6: Funciones y Eventos

Contacto: consultas@elearning-total.com
Web: www.elearning-total.com

Indice

Unidad 6: Funciones y Eventos

Objetivos

Que el alumno logre:

- Crear y aplicar funciones desarrolladas con Javascript

Introducción a las funciones

Cuando se desarrolla una aplicación compleja, es muy habitual utilizar una y otra vez las mismas instrucciones. Un script para una tienda de comercio electrónico por ejemplo, tiene que calcular el precio total de los productos varias veces, para añadir los impuestos y los gastos de envío.

Cuando una serie de instrucciones se repiten una y otra vez, se complica demasiado el código fuente de la aplicación, ya que:

- El código de la aplicación es mucho más largo porque muchas instrucciones están repetidas.
- Si se quiere modificar alguna de las instrucciones repetidas, se deben hacer tantas modificaciones como veces se haya escrito esa instrucción, lo que se convierte en un trabajo muy pesado y muy propenso a cometer errores.

Las funciones son la solución a todos estos problemas, tanto en JavaScript como en el resto de lenguajes de programación. Una función es un conjunto de instrucciones que se agrupan para realizar una tarea concreta y que se pueden reutilizar fácilmente.

¿Dónde insertar las funciones?

En principio, podemos colocar las funciones en cualquier parte de la página, siempre entre etiquetas <script type="text/javascript">.

No obstante existe una limitación a la hora de colocarla con relación a los lugares desde donde se la llame. Lo más sencillo es colocar la función antes de cualquier llamada a la misma y así seguro que nunca nos equivocaremos.

Existen dos opciones posibles para colocar el código de una función:

Colocar la función en el mismo bloque de script: En concreto, la función se puede definir en el bloque <script type="text/javascript"> donde esté la llamada a la función, aunque es indiferente si la llamada se encuentra antes o después del código de la función, dentro del mismo bloque.

```
<script type="text/javascript">
miFuncion()
function miFuncion(){
 //hago algo...
 document.write("Esto va bien")
}
</script>
```

Este ejemplo funciona correctamente porque la función está declarada en el mismo bloque que su llamada.

Colocar la función en otro bloque de script: También es válido que la función se encuentre en un bloque.

```
<!doctype html>
<html>
<head>
 <title>mi página</title>
<script type="text/javascript">
function mifuncion(){
 //hago algo...
 document.write("Hola Mundo!");
}
</script>
</head>
<body>
<script type="text/javascript">
mifuncion();
</script>
</body>
</html>
```

Parámetros de las funciones

Parámetros

Los parámetros se usan para mandar valores a las funciones.

Una función trabajará con los parámetros para realizar las acciones. Por decirlo de otra manera, los parámetros son los valores de entrada que recibe una función.

Por poner un ejemplo sencillo de entender, una función que realizase una suma de dos números tendría como parámetros a esos dos números.

Los dos números son la entrada, así como la salida sería el resultado de la suma.

Veamos una función para mostrar un mensaje de bienvenida en la página web, al que le vamos a pasar un parámetro que contendrá el nombre de la persona a la que hay que saludar.

```
function bienvenida(nombre) {
document.write("<h2>Hola " + nombre + "</h2>");
}
```

Como podemos ver en el ejemplo, para definir en la función un parámetro tenemos que poner el nombre de la variable que va a almacenar el dato que le pasemos.

Esa variable, que en este caso se llama nombre, tendrá como valor el dato que le pasemos a la función cuando la llamemos. Además, la variable donde recibimos el parámetro tendrá vida durante la ejecución de la función y dejará de existir cuando la función termine su ejecución.

Para llamar a una función que tiene parámetros se coloca entre paréntesis el valor del parámetro. Para llamar a la función del ejemplo habría que escribir:

```
bienvenida("Juan Perez");
```

Al llamar a la función así, el parámetro nombre toma como valor "Juan Perez" y al escribir el saludo por pantalla escribirá "Hola Juan Perez" entre etiquetas.

Los parámetros pueden recibir cualquier tipo de datos, numérico, textual, boleano o un objeto. Realmente no especificamos el tipo del parámetro, por eso debemos tener un cuidado especial al definir las acciones que realizamos dentro de la función y al pasarle valores, para asegurarnos que todo es consecuente con los tipos de datos que esperamos tengan nuestras variables o parámetros.

Múltiples parámetros

Una función puede recibir tantos parámetros como queramos y para expresarlo se colocan los nombres de los parámetros separados por comas, dentro de los paréntesis. Veamos rápidamente la sintaxis para que la función de antes, pero hecha para que reciba dos parámetros, el primero el nombre al que saludar y el segundo la profesión.

```
function bienvenida(nombre,profesion){
document.write("<h2>Hola " + nombre + "</h2>");
 document.write("<p>Profesión: " + profesion + "</p>");
}
```

Los parámetros se pasan por valor

Los parámetros de las funciones se pasan por valor. Esto quiere decir que estamos pasando valores y no variables. En la práctica, aunque modifiquemos un parámetro en una función, la variable original que habíamos pasado no cambiará su valor.

Se puede ver fácilmente con un ejemplo.

```
function pasoPorValor(miParametro){
miParametro = 58;
```

```
document.write("he cambiado el valor a 58");
}
var miVariable = 9;
pasoPorValor(miVariable);
document.write ("el valor de la variable es: " + miVariable);
```

En el ejemplo tenemos una función que recibe un parámetro y que modifica el valor del parámetro asignándole el valor 58.

También tenemos una variable, que inicializamos a 9 y posteriormente llamamos a la función pasándole esta variable como parámetro.

Como dentro de la función modificamos el valor del parámetro podría pasar que la variable original cambiase de valor, pero como los parámetros no modifican el valor original de las variables, ésta no cambia de valor. De este modo, una vez ejecutada la función, al imprimir en pantalla el valor de miVariable se imprimirá el número 9, que es el valor original de la variable, en lugar de 58 que era el valor con el que habíamos actualizado el parámetro.

En Javascript sólo se pueden pasar las variables por valor.

Valores de retorno

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.

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 del 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.

Funciones para cadenas de texto

Función length

Calcula la longitud de una cadena de texto (el número de caracteres que la forman)

```
var mensaje = "Hola Mundo";
var cantidadLetras = mensaje.length; // cantidadLetras = 10
```

Función +

Se emplea para concatenar varias cadenas de texto.

```
var mje1 = "Hola";
var mje2 = " Mundo";
var mjeCompleto = mje1 + mje2; // mjeCompleto = "Hola Mundo"
```

Función concat()

También se utiliza para concatenar datos.

```
var mje1 = "Hola";
var mje2 = mje1.concat(" Mundo"); // mje2 = "Hola Mundo"
```

Función toLowerCase()

Transforma todos los caracteres de la cadena a sus correspondientes caracteres en minúsculas.

```
var mje1 = "HolA";
var mje2 = mje1.toLowerCase(); // mje2 = "hola"
```

Función toUpperCase()

Transforma todos los caracteres de la cadena a sus correspondientes caracteres en mayúsculas.

```
var mje1 = "HolA";
var mje2 = mje1.toUpperCase(); // mje2 = "HOLA"
```

Función charAt(posicion)

Obtiene el carácter que se encuentra en la posición indicada.

```
var mje = "Hola";
```

```
var caracter = mje.charAt(0); // caracter = H
caracter = mje.charAt(2);     // caracter = l
```

Función indexOf(caracter)

Calcula la posición en la que se encuentra el carácter indicado dentro de la cadena de texto. Si el carácter se incluye varias veces dentro de la cadena de texto, se devuelve su primera posición empezando a buscar desde la izquierda. Si la cadena no contiene el carácter, la función devuelve el valor -1.

```
var mje = "Hola";
var posicion = mje.indexOf('o'); // posicion = 1
posicion = mje.indexOf('f');     // posicion = -1
```

Función lastIndexOf(caracter)

Calcula la última posición en la que se encuentra el carácter indicado dentro de la cadena de texto. Si la cadena no contiene el carácter, la función devuelve el valor -1

```
var mje = "Hola";
var posicion = mje.lastIndexOf('a'); // posicion = 3
posicion = mje.lastIndexOf('j');     // posicion = -1
```

La función lastIndexOf() comienza su búsqueda desde el final de la cadena hacia el principio, aunque la posición devuelta es la correcta empezando a contar desde el principio de la palabra.

Función substring(inicio, final)

Extrae una porción de una cadena de texto. El segundo parámetro es opcional. Si sólo se indica el parámetro inicio, la función devuelve la parte de la cadena original correspondiente desde esa posición hasta el final.

```
var mje = "Hola Mundo";
var extracto = mje.substring(3); // extracto = "a Mundo"
extracto = mje.substring(6);     // extracto = "undo"
extracto = mje.substring(8);     // extracto = "do"
```

Cuando se indica el inicio y el final, se devuelve la parte de la cadena original comprendida entre la posición inicial y la inmediatamente anterior a la posición final (es decir, la posición inicio está incluida y la posición final no).

```
var mje = "Hola Mundo";
```

```
var extracto = mje.substring(1, 8); // extracto = "ola Mun"
extracto = mje.substring(3, 4);     // extracto = "a"
```

Si se indica un final más pequeño que el inicio, JavaScript los considera de forma inversa, ya que automáticamente asigna el valor más pequeño al inicio y el más grande al final:

```
var mje = "Hola Mundo";
var extracto = mje.substring(5, 0); // extracto = "Hola "
extracto = mje.substring(0, 5);     // extracto = "Hola "
```

Función split(separador)

Convierte una cadena de texto en un array de cadenas de texto. La función parte la cadena de texto determinando sus trozos a partir del carácter separador indicado.

```
var mje = "Hola Mundo, bienvenido al curso de Javascript!";
var cadena = mje.split(" ");
// cadena = ["Hola", "Mundo,", "bienvenido", "al", "curso", "de", "Javascript"];
```

Con esta función se pueden extraer fácilmente las letras que forman una palabra:

```
var texto = "Javascript";
var letras = texto.split(""); // letras = ["J", "a", "v", "a", "s", "c", "r", "i", "p", "t"]
```

Funciones para arrays

Función lenght

Calcula el número de elementos de un array.

```
var letras = ["J", "a", "v", "a", "s", "c", "r", "i", "p", "t"];
var cantidadLetras = letras.length; // cantidadLetras = 10
```

Función concat()

Se emplea para concatenar los elementos de varios arrays.

```
var vector1 = [1, 2, 3];
vector2 = vector1.concat(4, 5, 6);   // vector2 = [1, 2, 3, 4, 5, 6]
vector3 = vector1.concat([4, 5, 6]); // vector3 = [1, 2, 3, 4, 5, 6]
```

Función join(separador)

Es la función contraria a split(). Une todos los elementos de un array para formar una cadena de texto. Para unir los elementos se utiliza el carácter separador indicado.

```
var vector = ["hola", "mundo"];
var mje = vector.join(""); // mje = "holamundo"
mje = vector.join(" ");    // mje = "hola mundo"
```

Función pop()

Elimina el último elemento del array y lo devuelve. El array original se modifica y su longitud disminuye en 1 elemento.

```
var vector = [1, 2, 3];
var ultimo = vector.pop();
// ahora vector = [1, 2], ultimo = 3
```

Función push()

Añade un elemento al final del array. El array original se modifica y aumenta su longitud en 1 elemento. (También es posible añadir más de un elemento a la vez).

```
var vector = [1, 2, 3];
vector.push(4);
// ahora vector = [1, 2, 3, 4]
```

Función shift()

Elimina el primer elemento del array y lo devuelve. El array original se ve modificado y su longitud disminuida en 1 elemento.

```
var vector = [1, 2, 3];
var primero = vector.shift();
// ahora vector = [2, 3], primero = 1
```

Función unshift()

Añade un elemento al principio del array. El array original se modifica y aumenta su longitud en 1 elemento. (También es posible añadir más de un elemento a la vez).

```
var vector = [1, 2, 3];
vector.unshift(0);
// ahora vector = [0, 1, 2, 3]
```

Función reverse()

Modifica un array colocando sus elementos en el orden inverso a su posición original.

```
var vector = [1, 2, 3];
vector.reverse();
// ahora vector = [3, 2, 1]
```

Funciones para números

Función NaN()

JavaScript emplea el valor NaN para indicar un valor numérico no definido (por ejemplo, la división 0/0).

```
var nro1 = 0;
var nro2 = 0;
alert(nro1/nro2); // se muestra el valor NaN
```

Función isNaN()

Permite proteger a la aplicación de posibles valores numéricos no definidos.

```
var nro1 = 0;
var nro2 = 0;
if(isNaN(nro1/nro2)) {
  alert("La división no está definida para los números indicados");
}
else {
  alert("La división es igual a => " + nro1/nro2);
}
```

Función Infinity()

Hace referencia a un valor numérico infinito y positivo (también existe el valor –Infinity para los infinitos negativos).

```
var nro1 = 10;
var nro2 = 0;
alert(nro1/nro2); // se muestra el valor Infinity
```

Función toFixed (dígitos)

Devuelve el número original con tantos decimales como los indicados por el parámetro dígitos y realiza los redondeos necesarios. Se trata de una función muy útil por ejemplo para mostrar precios.

```
var nro1 = 4564.34567;
nro1.toFixed(2); // 4564.35
nro1.toFixed(6); // 4564.345670
nro1.toFixed(); // 4564
```

Resumen

En esta Unidad…

Trabajamos con Funciones y Eventos

En la próxima Unidad…

Trabajaremos con formularios y validación