APRENDERAPROGRAMAR.COM

FUNCIONES JAVASCRIPT. CONCEPTO. PARÁMETROS O ARGUMENTOS Y TIPOS. PASO POR VALOR. RETURN. EJEMPLOS. (CU01122E)

Sección: Cursos
Categoría: Tutorial básico del programador web: JavaScript desde cero

Fecha revisión: 2029

Resumen: Entrega nº22 del Tutorial básico “JavaScript desde cero”.

Autor: César Krall

FUNCIONES JAVASCRIPT

Una función JavaScript es un fragmento de código que puede ser invocado para realizar tareas o devolver un resultado. Si has trabajado con otros lenguajes de programación el concepto te resultará familiar. Las funciones JavaScript son similares a lo que en otros lenguajes se denomina procedimientos, funciones o métodos.

Programar, usando un símil, podemos verlo como realizar un viaje por carretera. Cuando realizamos un viaje, aparte de la necesidad de definir el objetivo y estudiar la ruta del viaje (estructura del programa) podemos decir que: “En general, pero sobre todo para viajes complicados, conviene dividir el problema en subapartados”.

La estrategia del “divide y vencerás”... Es una de las estrategias más usadas en programación de ordenadores y, una vez más, abordaremos aquí el uso de esta filosofía compañera de viaje del programador. El concepto de función aplicado a la programación JavaScript es muy similar al aplicable a distintas facetas de la vida: un escritor divide su curso en capítulos y apéndices. Un profesor divide el contenido de la asignatura en temas. Un ingeniero divide el proyecto en partes como Memoria, Anejos, Pliego de Condiciones, Presupuesto y Planos. En una fábrica, organizan el trabajo dividiendo las áreas funcionales en recepción de materias primas, área de pre-proceso, área de proceso, área de post-proceso y área de carga y despacho de producto terminado.

De cara a la programación JavaScript, usaremos la división del código en funciones por ser una estrategia efectiva para resolver problemas complejos. Cada función será llamada para realizar su cometido en un orden establecido.

Además una función se puede llamar tantas veces como se desee, lo cual evita tener que repetir código y por otro lado permite que cuando haya que realizar una corrección únicamente tengamos que hacerla en la función concreta que se ve afectada.

Las funciones pueden recibir información para realizar su cometido, por ejemplo function suma (a, b) recibe dos elementos de información: a y b, o no recibirla por realizar un proceso que no necesita recibir información, por ejemplo function dibujarCirculo().

Otra característica interesante de las funciones es que permite abstraer los problemas. Supongamos que necesitamos una función que devuelva para un importe de una compra sin impuestos el importe con impuestos, y que a su vez el porcentaje de impuestos a aplicar depende del tipo de producto. Si un compañero nos facilita la función function obtenerImporteConImpuestos (importeSinImpuestos) no tenemos que preocuparnos del código de la función. Unicamente sabemos que invocando a la función obtendremos el importe con impuestos. De esta forma, podemos utilizar funciones que han creado otros programadores o funciones disponibles en librerías sin necesidad de conocer el código de las mismas. Decimos que las funciones son “cajas negras” que facilitan la abstracción porque no necesitamos ver en su interior, sólo nos interesan sus resultados.

De hecho, es posible que un programador use un código para una función function obtenerImporteConImpuestos (importeSinImpuestos) y otro programador use otro código para esa misma función sin que esto suponga ningún problema. Lo importante es que la función realice su cometido, no cómo lo realice ya que es frecuente que haya distintas maneras de hacer algo (aunque ciertamente hacer las cosas de diferente manera no debe significar que unas veces se hagan bien y otras mal: siempre deberían hacerse las cosas bien).

Una función en general debe tener un nombre descriptivo de cuál es su cometido y tener un cometido claro y único. No deben mezclarse tareas que no tengan relación entre sí dentro de una función.

FUNCIONES CON PARÁMETROS Y SIN PARÁMETROS

Una función JavaScript puede requerir ser llamada pasándole cierta información o no requerir información.

Definición de una función sin parámetros (no requiere información):

```
//Comentario descriptivo de qué hace la función
    function nombreDeLaFunción () {
//Código de la función
}
```

Definición de una función con parámetros (requiere información):

```
//Comentario descriptivo de qué hace la función
    function nombreDeLaFunción (param1, param2, …, paramN) {
//Código de la función
}
```

Una función puede recibir tantos parámetros como se deseen, aunque no sería demasiado razonable que una función reciba más de cuatro o cinco parámetros.

Los parámetros que se le pasan a la función pueden ser:

a) Valores simples a los que se denomina literales: por ejemplo 554, true ó ‘aldea’.

b) Variables que contienen un número, un texto o un valor booleano.

c) Objetos de naturaleza compleja, como arrays y otros tipos de objetos que veremos más adelante.

Cuando una función recibe un parámetro dicho parámetro funciona como si se tratara de una variable disponible para la función inicializada con el valor que se le pasa a la función.

Veamos un ejemplo:

```
function mostrarImporteConImpuestos(importeSinImpuestos) {
var importeConImpuestos; importeConImpuestos = importeSinImpuestos * 1.21;
msg = 'Importe antes de impuestos: '+ importeSinImpuestos + '\n\n';
alert(msg + 'Importe con impuestos: '+ importeConImpuestos + '\n\n');
}
```

Aquí vemos dos cosas de interés: el parámetro que recibe la función no tiene un tipo de datos explícito. El tipo de datos es "inferido" por el intérprete JavaScript.

Por otro lado, el parámetro está disponible dentro de la función con el valor con el que haya sido invocado. Por ejemplo onclick="mostrarImporteConImpuestos(100)" hará que importeSinImpuestos valga 100 porque ese es el valor con el que se invoca.

Cuando una función tiene varios parámetros, se debe invocar escribiendo su nombre seguido de los parámetros en el orden adecuado.

FUNCIONES QUE DEVUELVEN UN RESULTADO. RETURN.

Una función JavaScript puede devolver un resultado si se introduce la sentencia return resultado; donde resultado es aquello que queremos devolver (normalmente una variable que contiene un valor numérico, de texto o booleano, pero también podrían ser objetos con mayor complejidad como un array).

Una vez se llega a la sentencia return se produce la devolución del resultado y se interrumpe la ejecución de la función. Por ello la sentencia return será normalmente la última instrucción dentro de una función.

Definición de una función sin parámetros que devuelve un resultado:

```
//Comentario descriptivo de qué hace la función
    function nombreDeLaFunción () {
//Código de la función
return resultado;
}
```

Definición de una función con parámetros que devuelve un resultado:

```
//Comentario descriptivo de qué hace la función
    function nombreDeLaFunción (param1, param2, …, paramN) {
//Código de la función
return resultado;
}
```

Una función sólo devolverá un resultado y normalmente sólo tendrá una sentencia return, aunque si hay sentencias condicionales como if, puede haber varias sentencias return: una sentencia return para cada sentencia condicional.

Si además del resultado la función incluye código que implique acciones como mostrar un mensaje por pantalla, se ejecutará el código a la vez que se devuelve el resultado.

Veamos un ejemplo:

```
function obtenerImporteConImpuestos(importeSinImpuestos) {
var importeConImpuestos; importeConImpuestos = importeSinImpuestos * 1.21;
return importeConImpuestos;
}
```

Un ejemplo de uso de esta función sería:

onclick="alert('Calculado lo siguiente para producto de precio 100: importe con impuestos vale ' + obtenerImporteConImpuestos(100));"

Aquí vemos cómo al invocar la función ésta devuelve un resultado que se coloca allí donde se encuentra la llamada a la función (en este ejemplo el resultado se coloca dentro de una sentencia para mostrar un mensaje por pantalla. Aquí el resultado es numérico, pero el intérprete lo transformará automáticamente en texto para mostrarlo por pantalla).

También será frecuente almacenar el resultado en una variable, por ejemplo: var importeConImp = obtenerImporteConImpuestos(100);

El resultado que devuelve una función puede ser:

a) Un valor simple (literal): por ejemplo 554, ó true ó 'aldea'.

b) Una variable que contienen un número, un texto o un valor booleano.

c) Un objeto de naturaleza compleja, como arrays y otros tipos de objetos que veremos más adelante.

El resultado que devuelve una función no tiene un tipo de datos explícito. El tipo de datos es "inferido" por el intérprete JavaScript.

LLAMADAS A FUNCIONES DESDE OTRAS FUNCIONES

Una función puede llamar a otra función simplemente escribiendo su nombre y los parámetros que sean necesarios. Ejemplo:

```
function mostrarImporteConImpuestos2(importeSinImpuestos) {
var msg; msg = 'Ejemplo. Importe antes de impuestos: '+ importeSinImpuestos + '\n\n';
alert(msg + 'Importe con impuestos: '+ obtenerImporteConImpuestos(importeSinImpuestos) + '\n\n');
}
```

En esta función en vez de realizarse el cálculo del importe con impuestos, se invoca otra función que es la que se encarga de realizar el cálculo y devolver el valor correspondiente.

PASO DE PARÁMETROS A FUNCIONES

Hay dos formas comunes de pasar parámetros a funciones en programación: por valor, que implica que si se pasa una variable sus cambios sólo son conocidos dentro de la función, o por variable, que implica que si se pasa una variable ésta puede ser modificada por la función y sus cambios ser conocidos fuera de la función. JavaScript trabaja con paso de parámetros por valor, lo que implica que la variable pasada como parámetro funciona como una variable local a la función: si el parámetro sufre cambios, estos cambios sólo son conocidos dentro de la función. La variable “verdadera” no puede ser modificada.

PASO DE UN NÚMERO DE PARÁMETROS INCORRECTO

Si se pasan más parámetros de los necesarios, JavaScript ignorará los parámetros sobrantes. Si se pasan menos parámetros de los necesarios, JavaScript asignará valor undefined a los parámetros de los que no se recibe información y se ejecutará sin que surja ningún mensaje de error (aparte de los posibles resultados extraños que esto pudiera ocasionar).

EJERCICIOS

1. Crea un script donde declares una función obtenerImporteConImpuestos que reciba dos parámetros: el importe sin impuestos (numérico) y el tipo de producto (numérico entero). La función debe mostrar por pantalla el importe sin impuestos más el 21% si el tipo de producto es 1, ó el importe sin impuestos más el 10% si el tipo de producto es 2, ó el importe sin impuestos más el 5% si el tipo de producto es 3.

Ejemplo: obtenerImporteConImpuestos(100, 1) debe mostrar: Para un importe sin impuestos de 100 y tipo de producto 1 el resultado de importe con impuestos es 121. obtenerImporteConImpuestos(100, 2) debe mostrar: Para un importe sin impuestos de 100 y tipo de producto 2 el resultado de importe con impuestos es 110.

2. Crea un script donde declares una función obtenerImporteConImpuestos2 que reciba un parámetro: el importe sin impuestos (numérico). La función debe devolver un array con valor undefined para el índice 0, el importe sin impuestos más el 21% para el índice 1, el importe sin impuestos más el 10% para el índice 2, ó el importe sin impuestos más el 5% para el índice 3. Invoca la función haciendo que se muestre el contenido del array por pantalla.

obtenerImporteConImpuestos(100) debe devolver: resultado[0] = undefined, resultado[1] = 121, resultado[2] = 110, resultado[3] = 105. Por pantalla se debe mostrar: Para precio sin impuestos 100 si el producto es tipo 1 el importe es 121, si el producto es tipo 2 el importe es 110 y si el producto es tipo 3 el importe es 105.

Ejecuta el código y comprueba sus resultados. Para comprobar si es correcta tu solución puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU01123E

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=78&Itemid=206