UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Programador Web Inicial
Front End Developer

JS – Función Retorno de Variables

Universidad Tecnológica Nacional - Facultad Regional Buenos Aires
Medrano 951 (C1179AAQ) C.A.B.A. Tel: (54 11) 4867-7511

Una función retorna valores

Algunas funciones no devuelven un valor significativo después de su finalización, pero otras sí, y es importante comprender cuáles son sus valores, cómo utilizarlos en su código y cómo hacer que sus propias funciones personalizadas devuelvan valores útiles.

¿Qué son los valores de retorno?

Los valores de retorno son exactamente como suenan: los valores devueltos por la función cuando se completa. Ya has alcanzado los valores de retorno varias veces, aunque es posible que no hayas pensado en ellos explícitamente. Volvamos a un código familiar:

```
var myText = 'I am a string';
var newString = myText.replace('string', 'sausage');
console.log(newString);
// la función de cadena replace () toma una cadena,
// sustituyendo una subcadena con otra y devolviendo
// una cadena nueva con la sustitución realizada
```

Vimos exactamente este bloque de código en nuestro primer artículo de función. Estamos invocando la función `replace ()` en la cadena `myText`, y le pasamos dos parámetros: la subcadena a encontrar y la subcadena con la que reemplazarla.

Cuando esta función se completa (termina de ejecutarse), devuelve un valor, que es una nueva cadena con el reemplazo realizado. En el código anterior, estamos guardando este valor de retorno como el valor de la variable `newString`.

Algunas funciones no devuelven un valor de retorno como tal. Por ejemplo, en la función `displayMessage ()` que creamos anteriormente, no se devuelve ningún valor específico como resultado de la función que se invoca. Simplemente hace que aparezca un cuadro en algún lugar de la pantalla, ¡eso es todo!

Generalmente, se usa un valor de retorno donde la función es un paso intermedio en un cálculo de algún tipo. Quieres llegar a un resultado final, que involucra algunos valores. Esos valores deben ser calculados por una función, que luego devuelve los resultados para que puedan usarse en la siguiente etapa del cálculo.

Uso de valores de retorno en sus propias funciones

Para devolver un valor de una función personalizada, debe usar la palabra clave `return`. La funcion `draw()` dibuja 100 círculos aleatorios en algún lugar del `<canvas>`

```
function draw() {
  ctx.clearRect(0,0,WIDTH,HEIGHT);
  for (var i = 0; i < 100; i++) {
    ctx.beginPath();
    ctx.fillStyle = 'rgba(255,0,0,0.5)';
    ctx.arc(random(WIDTH), random(HEIGHT), random(50), 0, 2 * Math.PI);
    ctx.fill();
  }
}
```

Dentro de cada iteración del bucle, se realizan tres llamadas a la función `random()` para generar un valor aleatorio para la **coordenada x**, **la coordenada y** y **el radio del círculo actual**, respectivamente.

La función `random()` toma un parámetro, un número entero, y devuelve un número aleatorio entero entre 0 y ese número. Se parece a esto:

```
function randomNumber(number) {
  return Math.floor(Math.random()*number);
}
```

Esto podría escribirse de la siguiente manera:

```
function randomNumber(number) {
  var result = Math.floor(Math.random()*number);
  return result;
}
```

Pero la primera versión es más rápida de escribir y más compacta.

Devolvemos el resultado del cálculo `Math.floor(Math.random()*number)` cada vez que se llama a la función. Este valor de retorno aparece en el punto en que se llamó a la función y el código continúa. Entonces, por ejemplo, si ejecutamos la siguiente línea:

```
ctx.arc(random(WIDTH), random(HEIGHT), random(50), 0, 2 * Math.PI);
```

y las tres llamadas `random()` devolvieron los valores 500, 200 y 35, respectivamente, la línea en realidad se ejecutaría como si fuera esto:

```
ctx.arc(500, 200, 35, 0, 2 * Math.PI);
```

Las llamadas de la función se ejecutan primero y sus valores de retorno se sustituyen por la función, antes de que se ejecute la línea en sí.

Aprendizaje activo: nuestra propia función de valor de retorno

Probemos escribir nuestras propias funciones con valores de retorno.

1. En primer lugar, haga una copia local del archivo function-library.html de GitHub. Esta es una página HTML simple que contiene un campo de texto `<input>` y un párrafo. También hay un elemento `<script>` en el que hemos almacenado una referencia a ambos elementos HTML en dos variables. Esta pequeña página le permitirá ingresar un número en el cuadro de texto y mostrar diferentes números relacionados con él en el párrafo a continuación.
2. Agreguemos algunas funciones útiles a este elemento <script>. Debajo de las dos líneas existentes de JavaScript, agregue las siguientes definiciones de funciones:

```
function squared(num) {
  return num * num;
}

function cubed(num) {
  return num * num * num;
}

function factorial(num) {
  var x = num;
  while (x > 1) {
    num *= x-1;
    x--;
  }
  return num;
}
```

Las funciones `squared()` y `cubed()`: devuelven el cuadrado o el cubo del número dado como parámetro. La función `factorial()` devuelve el factorial del número dado.

A continuación, vamos a incluir una forma de imprimir información sobre el número ingresado en la entrada de texto. Introduzca el siguiente controlador de eventos debajo de las funciones existentes:

```
input.onchange = function() {
  var num = input.value;
  if (isNaN(num)) {
    para.textContent = 'You need to enter a number!';
  } else {
    para.textContent = num + ' squared is ' + squared(num) + '. ' +
                       num + ' cubed is ' + cubed(num) + '. ' +
                       num + ' factorial is ' + factorial(num) + '.';
  }
}
```

3. Aquí estamos creando un controlador de eventos `onchange` que se ejecuta cada vez que el evento de cambio se activa en la entrada de texto, es decir, cuando se ingresa un nuevo valor en la entrada de texto y se envía (ingrese un valor y luego presione la pestaña, por ejemplo). Cuando se ejecuta esta función anónima, el valor existente ingresado en la entrada se almacena en la variable `num`

   A continuación, hacemos una prueba condicional: si el valor ingresado no es un número, imprimimos un mensaje de error en el párrafo. La prueba analiza si la expresión `isNaN(num)` devuelve verdadero. Usamos la función `isNaN()` para probar si el valor num no es un número; si es así, devuelve verdadero, y si no, falso.

   Si la prueba devuelve falso, el valor numérico es un número, por lo que imprimimos una oración dentro del elemento de párrafo que indica cuál es el cuadrado, el cubo y el factorial del número. La oración llama a las funciones `squared(), cubed()` y `factorial()` para obtener los valores requeridos.

4. Guarde su código, cárgalo en un navegador y pruébelo.