Introducción a JavaScript

Francisco Javier Arce Anguiano

Introducción a JavaScript **1**
5. Funciones en JavaScript 3
5.1. Funciones en JavaScript 3
5.2. Parámetros en las funciones 4
5.3. Regreso de valores en una función con la sentencia return 5
5.4. Variables locales y globales en las funciones 6
5.5. Funciones recursivas 6
5.6. Entrada y validación del factorial 8
5.7. Funciones anidadas 9
5.8. Usar el objeto arguments 10
5.9. Parámetros predeterminados 11

5. Funciones en JavaScript

Objetivo: El alumno aprenderá a crear funciones y estructurar sus programas a base de ellas.

5.1. Funciones en JavaScript

Las funciones son uno de los pilares en los que se basa JavaScript. Una función es un conjunto de sentencias JavaScript que realizan alguna tarea específica. Las partes que definen una función son:

- El nombre de la función.
- La lista de argumentos de la función, encerrados entre paréntesis y separados por comas(“,”).
- Las sentencias en JavaScript que definen la función, encerradas entre llaves({,});

Una función puede incluir llamadas a otras funciones definidas en la aplicación, pero únicamente de la página actual. Una opción interesante es definir las funciones en el encabezado del documento de manera que, cuando se inicialice la página, las funciones se definan antes de cualquier acción del usuario. La sintaxis de definición de una función sería:

```
function nombreFuncion(parametro1, parametro2...){
       Instrucciones
}
```

La definición de una función no implica su ejecución; ésta sólo se ejecutará cuando se le realice la llamada pertinente.

Ejemplo: Se define una función llamada saludo que será posteriormente llamada para mostrar la cadena de texto especificada.

```
<html>
<head>
<title>Ejemplo de funciones</title>
<script language="javascript">
function saludo()
{
    document.write("Hola amigos, esto es un saludo");
}
 </script>
 </head>
<body>
<script language="javascript">
    saludo();
 </script>
</body>
</html>
```

5.2. Parámetros en las funciones

Los argumentos de una función permiten que el resultado varíe según el valor indicado de la misma. Estos pueden ser variables, números u objetos.

En el siguiente script, podrá ver el método de definición de los argumentos de la función y posteriormente, el modo de llamar a cada argumento.

```
<html>
<head>
<title>Ejemplo de funciones</title>
<script language="javascript">
function Mensaje(Respuesta)
{
   if(Respuesta==0)  alert("La respuesta es correcta");
   if(Respuesta==2)  alert("La respuesta es incorrecta.  Repasa la
leccion 10");
   if(Respuesta==1)  alert("Vaya disparate. Debes repetir curso");
   if(Respuesta>=3)  alert("Respuesta fuera de rango");
}
```

```
 </script>
 </head>
<body>
<script language="javascript">
Mensaje(0);
Mensaje(1);
Mensaje(2);
Mensaje(5);
 </script>
</body>
</html>
```

5.3. Regreso de valores en una función con la sentencia return

Para que una función devuelva el resultado de una serie de operaciones procedentes de la misma función, utilizaremos la instrucción return.

```
<html>
<head>
<title>Ejemplo de funciones</title>
<script language="javascript">
function mitad(valor)
{
   return valor/2;
}
 </script>
 </head>
<body>
<script language="javascript">
document.write("La mitad de 100 es"+mitad(100));
</script>
</body>
</html>
```

5.4. Variables locales y globales en las funciones

Las variables pueden ser globales y locales según el lugar en que se hayan definido. Las primeras pueden usarse en cualquier parte del código, mientras que las segundas sólo pueden hacerlo dentro de la función que las define.

```
 //Definicion de variable global
 var variable1="Esta cadena de texto se ha definido FUERA de una
funcion. Es global";
 //Definicion de una variable local
 function ejemplo()
 {
    var variable2="Esta cadena de texto se ha definido DENTRO de una
funcion. Es local";
    document.write(variable2);
 }
 //Impresion en pantalla de la variable global
 document.write(variable1+"<br>");
 //Impresion en pantalla de la variable local
 ejemplo()
```

Si se intenta utilizar una variable local en un ámbito global, JavaScript dará un mensaje de error diciendo que la variable no está definida.

5.5. Funciones recursivas

- Una función se puede llamar a sí misma.
- A ese proceso lo conocemos como recursividad.
- Es muy importante crear una sentencia condicional para detener la recursividad.

$$n! = n \times (n-1) \times (n-2) \times ... \times 1$$

$$1! = 1$$
$$2! = 2 \times 1 = 2$$
$$3! = 3 \times 2 \times 1 = 6$$
$$4! = 4 \times 3 \times 2 \times 1 = 24$$
$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

```
<!DOCTYPE html>
<html>
<head>
	<meta charset="utf-8">
	<title>Factorial</title>
	<script>
		function factorial(n) {
			//Criterio de finalización
			if ((n===0) || (n===1)) {
				return 1;
			} else {
				return (n * factorial(n-1));
			}
		}
		console.log(5, factorial(5));
		console.log(8, factorial(8));
		console.log(12, factorial(12));
	</script>
</head>
<body>
</body>
```

```
</html>
```

5.6. Entrada y validación del factorial

- Crearemos la entrada de datos.
- Validaremos la entrada.
- Formatearemos la salida.

```
<!DOCTYPE html>
<html>
<head>
	<meta charset="utf-8">
	<meta name="viewport" content="width=device-width, initial-scale=1">
	<title>Factorial</title>
	<script>
		window.onload = function() {
document.getElementById("calcular").addEventListener("click",function(){
			let n = document.getElementById("numero").value;
			n = Math.abs(parseInt(n));
			if (isNaN(n) || n > 100) {
				return;
			}
			f = factorial(n);
			salida = n.toString()+"! = "+f;
			document.getElementById("salida").innerHTML = salida;
		});
		}
		function factorial(n) {
			//Criterio de finalización
			if ((n===0) || (n===1)) {
				return 1;
			} else {
				return (n * factorial(n-1));
			}
		}
		//console.log(5, factorial(5));
		//console.log(8, factorial(8));
		//console.log(12, factorial(12));
	</script>
</head>
<body>
	<label for="numero">Introduzca un número entero positivo menor a
```

```
100:</label>
      <input type="text" name="numero" id="numero">
      <button type="button" id="calcular">Calcula factorial</button>
      <p id="salida"> </p>
</body>
</html>
```

5.7. Funciones anidadas

- Podemos escribir una función dentro de otra función.
- Las funciones internas, sólo podrán ser accedidas desde la función contenedora, pero no fuera de ésta.
- Las variables, constantes y demás elementos de la función contenedora (padre) pueden ser accedidos desde las funciones internas (hijos), pero no al revés.

```
<!DOCTYPE html>
<html>
<head>
     <meta charset="utf-8">
     <title>Funciones anidadas</title>
     <script>
          function sumaCuadrados(a, b) {
               function cuadrado(x){
                    return x * x;
               }
               return cuadrado(a) + cuadrado(b);
          }
          console.log(sumaCuadrados(3,4));
          //console.log(cuadrado(5));
     </script>
</head>
<body>

</body>
</html>
```

5.8. Usar el objeto arguments

- El objeto arguments de una función se mantiene en un objeto similar a un arreglo.
- Dentro de una función, puedes abordar los argumentos que se le pasan de la siguiente manera:
- arguments[i]
- donde i es el número ordinal del argumento, comenzando en 0.
- El número total de argumentos se indica mediante arguments.length.

```
<!DOCTYPE html>
<html>
<head>
     <title>Funciones | Arguments</title>
     <script>
          function suma() {
               var total = 0;
               for (var i = 0; i < arguments.length; i++) {
                    total += arguments[i];
               }
               return total;
          }
          function concatenar(separador) {
               var cadena = "";
               for (var i = 1; i < arguments.length; i++) {
                    cadena += arguments[i]
                    if (i < arguments.length-1) {
                         cadena += separador+" ";
                    }
               }
               return cadena;
          }
          console.log(suma());
          console.log(suma(4));
          console.log(suma(4,5));
          console.log(suma(4,5,6));
          console.log(suma(4,5,7,8,9,2,3,4,5));
          //
```

```
          console.log(concatenar(",","Pedro","Juan","Pablo"));
          console.log(concatenar("
*","Pedro","Juan","Pablo","Toño","Pepe"));
     </script>
</head>
<body>

</body>
</html>
```

5.9. Parámetros predeterminados

- Antes del ES6 o ECMA 2015 validamos los parámetros como undefined.
- Después del ES6 ya tenemos parámetros predefinidos.

```
<!DOCTYPE html>
<html>
<head>
     <title>Funciones | Parámetro </title>
     <script>
          function antesECMA2015(a, b) {
               b = typeof b !== 'undefined' ? b : 1;
               return a * b;
          }
          function despuesECMA2015(a, b = 1) {
               return a * b;
          }
          console.log(antesECMA2015(5));
          console.log(antesECMA2015(5,2));
          console.log(despuesECMA2015(3));
          console.log(despuesECMA2015(3,4));
     </script>
</head>
<body>
</body>
</html>
```