cedia | TMO Training Management Office

info@cedia.org.ec
cedia.edu.ec

CURSO DE JAVASCRIPT

Corporación Ecuatoriana para el Desarrollo de la Investigación y la Academia

Ing. María Fernanda Granda J., PhD

11-11-2025

“Conectando ideas, transformando sociedades”

Módulo 2: Funciones

- ¿Qué es una Función en JavaScript?
- Declaración de Funciones
- Definición de Parámetros y Argumentos
- Parámetros de las Funciones
- Valores de Retorno
- Parámetros obligatorios y opcionales
- Formas de Definir una función
- ¿Qué es la escritura Camel Case en JavaScript?
- Funciones predefinidas en JavaScript

¿Qué es una Función en JavaScript?

- Es un fragmento de código que puede ser invocado para realizar tareas o devolver un resultado.
- Puede, o no, recibir parámetros/valores y también puede, o no, devolver un resultado.
- ¿por qué utilizarlas?
  - Reutilización de código
  - Organización y claridad

Declaración de Funciones

Sintaxis básica:

```javascript
function saludar() {
  console.log("¡Hola!");
}
```

- Luego entre las llaves indicaremos todas las instrucciones de la función.

Definición de Parámetros y Argumentos

- Parámetros en la función y argumentos en la llamada.

```
function saludar(nombre){
    alert("Hola "+nombre);
}

saludar("Ma Fernanda");
saludar("Noralma");
saludar("Gloria");
```

Parámetros de las Funciones

- Pueden ser 0 o más parámetros.

```
function saludar(nombre, apellidos){
    alert("Hola "+ nombre +" "+ apellidos);
}

saludar("Ma Fernanda", "Granda");
saludar("Noralma", "Pincay");
saludar("Gloria"+"Segura");
```

Valores de Retorno

- Las funciones pueden **devolver un resultado**.
- La sentencia **return** finaliza la ejecución de la función y especifica un valor para ser devuelto.

```
function sumar(numero1, numero2) {
 return numero1 + numero2;
}
```

  - let resultado=sumar(40,18);
  - alert("40 + 18 es igual a "+resultado);
- El resultado se puede guardar en una variable

Valores de Retorno

- O usarlo directamente.

```
function sumar(numero1, numero2) {
 return numero1 + numero2;
}
```

  - alert("40 + 18 es igual a "+sumar(40,18));

Parámetros obligatorios y opcionales

- Parámetros con valores por defecto:

```
function saludar(nombre="visitante"){
    alert("Hola "+ nombre);
}

saludar();
saludar("Gloria")
```

A practicar!

- Crear una función que reciba los parámetros: nombres, apellidos y país de origen y que devuelva un saludo:
- Hola, nombre + apellido, eres del país de Origen.
- Usar Ecuador como valor por defecto.
- Ejecutarla.

Formas de Definir una función

Por declaración

- La forma más común de crear las funciones
- Se puede ejecutar la función incluso antes de haberla creado. Es decir es compatible con el **hoisting**, característica de JavaScript que, primero busca las declaraciones de funciones y luego procesa el código.

```
saludar(); // Hola

function saludar(){
  return "Hola";
}
```

Formas de Definir una función

Por expresión

- Permite guardar una función dentro de una variable, para luego ejecutar dicha variable.
- Sólo están disponibles a partir de la inicialización de la variable.

```
const saludo = function saludar() {
  return "Hola";
};

saludo(); // 'Hola'
```

Formas de Definir una función

Anónima o Lambda

- Se declaran sin nombre de función y se alojan en el interior de una variable, haciendo referencia a ella cada vez que queramos utilizarla.

```
// Función anónima "saludo"
const saludo = function() {
  return "Hola";
};

saludo(); // 'Hola'
```

Formas de Definir una función

Funciones Flecha

- Son una forma corta de escribir funciones que aparece en JavaScript a partir de ES6. Básicamente, se trata eliminar la palabra function y añadir => antes de abrir las llaves:

```
const func = function () {
  return "Función tradicional";
};

const func = () => {
  return "Función flecha";
};
```

Formas de Definir una función

Funciones Flecha - Ventajas

- Si la función solo tiene una línea, se omite las llaves {}
- En ese caso, automáticamente se hace un return de esa única línea, por lo que podemos omitir también el return.
- Si no tiene parámetros, se indica: () ⇒
- Si solo tiene un parámetro, se indica simplemente con el nombre del mismo: e ⇒
- Si tiene 2 o más parámetros, se indican: (a, b) ⇒

```
const func = () => "Función flecha."; // 0 parámetros: Devuelve "Función flecha"
const func = (e) => e + 1; // 1 parámetro: Devuelve el valor de e + 1
const func = (a, b) => a + b; // 2 parámetros: Devuelve el valor de a + b
```

Formas de Definir una función

Funciones Autoejecutables IIFE

- Se tiene que envolver entre paréntesis la función anónima y luego, agregar otro par de paréntesis para ejecutarla.

```
// Función autoejecutable
(function () {
  console.log("Hola!!");
})(); // Hola!!

// Función autoejecutable con parámetros
(function (name) {
  console.log(`¡Hola, ${name}!`);
})("Ale"); // ¡Hola, Ale!
```

Formas de Definir una función

Funciones Closure o Clausuras

- Se define como una función que “encierra” variables en su propio ámbito (y que continúan existiendo aun habiendo terminado la función).

```
// Clausura: Función incrementar()
const incrementar = (function () {
  let num = 0;
  return function () {
    num++;
    return num;
  };
})();

typeof incrementar; // 'function'
incrementar(); // 1
incrementar(); // 2
incrementar(); // 3
```

Formas de Definir una función

Funciones CallBacks

- Sirve pasar una “función B” por parámetro a una “función A”, de modo que la función A puede ejecutar esa función B de forma genérica desde su código.

```
const funcionB = function(){
        console.log("Función B ejecutada.");
};

const funcionA = function(callback){
        callback();
};

funcionA(funcionB); // Función B ejecutada.
```

¿Qué es la escritura Camel Case en JavaScript?

- Es una forma de separar varias palabras de una variable o función.
- La primera palabra empieza con minúscula y luego se escribe la primera letra de cada palabra en mayúscula y sin espacios. Ejemplos:
- let primerNombre=“Maria”;
- parseInt();

Funciones predefinidas en JavaScript

- Son funciones globales, que se llaman globalmente sin hacer referencia a un objeto
- Algunas ya conocidas:
  - alert()
  - prompt()
  - confirm()
- Hacen referencia a un objeto
  - console.log()

Funciones predefinidas en JavaScript

- Otras nuevas y muy útiles
  - parseInt()
  - parseFloat()
  - toFixed()
  - isNaN()
  - isFinite()
  - encodeURI()
  - decodeURI()
  - encodeURIComponent()
  - decodeURIComponent()
  - eval()

Funciones predefinidas en JavaScript

parseInt()

- toma un valor e intenta transformarlo en un número entero
- Si falla devuelve NaN
- **Ejemplos en Consola:**

```
> parseInt('123')
< 123
> parseInt('abc123')
< NaN
> parseInt('1abc23')
< 1
> parseInt('123abc')
< 123
```

Funciones predefinidas en JavaScript

parseInt()

- Admite un segundo argumento opcional que indica la base del número que se le está pasando (decimal, hexadecimal, binario, etc).
- Se recomienda especificar la base para evitar problemas de interpretación.

```
> parseInt(" 0xF", 16);
< 15
> parseInt(" F", 16);
< 15
> parseInt("17", 8);
< 15
> parseInt(021, 8);
< 15
> parseInt("015", 10);
< 15
> parseInt(15.99, 10);
< 15
> parseInt("FXX123", 16);
< 15
> parseInt("1111", 2);
< 15
> parseInt("15*3", 10);
< 15
> parseInt("15e2", 10);
< 15
> parseInt("15px", 10);
< 15
> parseInt("12", 13);
< 15
```

Funciones predefinidas en JavaScript

parseFloat()

- toma un valor e intenta transformarlo en un número de coma flotante (con decimales)
- **Ejemplos en Consola:**

```
> parseFloat('123')
< 123
> parseFloat('1.23')
< 1.23
> parseFloat('1.23abc.00')
< 1.23
> parseFloat('a.bc1.23')
< NaN
```

Funciones predefinidas en JavaScript

isNaN()

- Comprueba si el valor que se le pasa es un número válido.
- Devuelve true en caso de que no lo sea.
- **Ejemplos en Consola:**

```
> isNaN(NaN)
< true
> isNaN(123)
< false
> isNaN(1.23)
< false
> isNaN(parseInt('abc123'))
< true
```

Funciones predefinidas en JavaScript

isFinite()

- Comprueba si el valor que se le pasa es un número no es ni Infinity ni NaN.
- **Ejemplos en Consola:**

```
> isFinite(Infinity)
< false
> isFinite(-Infinity)
< false
> isFinite(12)
< true
> isFinite(1e308)
< true
> isFinite(1e309)
< false
```

Funciones predefinidas en JavaScript

encodeURI()

- Nos permite escapar (codificar) una URL reemplazando algunos caracteres por su correspondiente secuencia de escape UTF-8
- encodeURI nos devuelve una URI usable (sólo codifica algunos caracteres)
- **Ejemplo en Consola:**

```
> var url = 'http://www.packtpub.com/scr ipt.php?q=this and that';
> encodeURI(url);
< http://www.packtpub.com/scr%20ipt.php?q=this%20and%20that
```

decodeURI()

- Nos permite decodificar) un string codificado por encodeURI().

Funciones predefinidas en JavaScript

encodeURIComponent() y decodeURIComponent()

- Lo mismo que encodeURI() pero esta función codifica (decodifica) TODOS los caracteres transformables.
- **Ejemplo en Consola:**

```
> encodeURIComponent(url);

< "http%3A%2F%2Fwww.packtpub.com%2Fscr%20ipt.php%3Fq%3Dthis%20and%20that"
```

Funciones predefinidas en JavaScript

eval()

- Toma una cadena de texto y la ejecuta como código JavaScript.
- No debe usarse básicamente por dos motivos:
- **Rendimiento**: es mucho más lento evaluar código en vivo que tenerlo directamente en el script.
- **Seguridad**: Teniendo en cuenta que ejecuta todo lo que se le pase puede ser un agujero de seguridad.
- **Ejemplo por consola:**

```
> eval('var ii = 2;')
> ii
< 2
```

Funciones predefinidas en JavaScript

- Otras nuevas y muy útiles
  - Globales
    - toFixed()
    - parseInt()
    - parseFloat()
    - isNaN()
  - Hacen referencia a un objeto:
    - Math.sqrt()
    - Math.round()
    - Math.random()

cedia | TMO Training Management Office

GRACIAS

cedia.edu.ec
info@cedia.org.ec
(+593) 7-407-9300