Programador JavaScript Avanzado

Unidad 1: Conceptos principales de JS

Contactos: tecnologia@elearning-total.com
Web: www.elearning-total.com

Indice

Unidad 1: Conceptos principales de JS

- Funciones lambda o flecha

Objetivos

Que el alumno logre:

- Crear y aplicar funciones desarrolladas con Javascript

Funciones en Javascript

Las funciones son uno de los **bloques de construcción fundamentales en JavaScript**.

Una función en JavaScript es similar a un procedimiento — un conjunto de instrucciones que realiza una tarea o calcula un valor, pero para que un procedimiento califique como función, debe tomar alguna entrada y devolver una salida donde hay alguna relación obvia entre la entrada y la salida. Para usar una función, es necesario definirla en algún lugar del ámbito desde el que va a hacer llamada.

Sintaxis

La forma clásica de declarar funciones es a través de la palabra reservada function, seguida del nombre de la función y sus parámetros entre paréntesis.

El cuerpo de la función, que es el código que realiza la tarea específica, está delimitado por llaves

```
function  saludar () {
        alert('Hola mundo!')
}
```

Para llamar a una función en JavaScript, se utiliza el nombre de la función seguido de los parámetros entre paréntesis. Por ejemplo:

```
<a onclick="saludar()">Saludo</a>
```

Las funciones en JavaScript pueden ser asignadas a variables, pasadas como argumentos a otras funciones, devueltas como valores de otras funciones y también pueden ser anidadas dentro de otras funciones.

Además de las funciones definidas con la palabra clave "function", en JavaScript también existen las funciones flecha (arrow functions) o funciones lambda, que son una forma más concisa de definir funciones y que se utilizan con frecuencia en el código moderno de JavaScript.

Funciones Lambda o funciones flecha

Las funciones lambda (también conocidas como funciones anónimas) son funciones sin nombre que se pueden definir en una sola línea de código.

Estas funciones son comúnmente utilizadas en lenguajes de programación funcionales y permiten la creación de funciones simples y rápidas sin la necesidad de definir una función completa con un nombre.

Las funciones lambda son útiles en situaciones donde se requiere una función simple y rápida, como ordenar una lista o en la definición de funciones de filtro. También se utilizan a menudo en la programación de funciones de alto nivel en la que se pasan funciones como argumentos a otras funciones.

En JavaScript, las funciones lambda se conocen como funciones flecha (arrow functions) y se definen utilizando la sintaxis de flecha (=>).

La sintaxis básica de una función flecha es la siguiente:

```
(parametro1, parametro2, ...) => expresion
```

Los parámetros son los argumentos de la función y la expresión es el cuerpo de la función.

Por ejemplo, la siguiente función flecha toma dos argumentos y devuelve un resultado:

```
const multiplicar = (3, 2) => 3*2
```

Las funciones flecha también pueden tener cuerpos de función más complejos y pueden incluir múltiples líneas de código. En ese caso, se debe utilizar la sintaxis de llaves para delimitar el cuerpo de la función y la palabra clave "return" para devolver un valor. Por ejemplo:

```
const multiplicar = (3) => {
  const resultado = 3 * 2;
  return resultado;
};
```

Paso a paso

Paso a paso, la descomposición de una "función tradicional" hasta la "función flecha" más simple:

```
// Función tradicional
function (a){
  return a + 100;
}
```

```
// Desglose de la función flecha
// 1. Eliminar la palabra "function" y colocar la flecha entre el argumento y el corchete de apertura.
(a) => {
  return a + 100;
}

// 2. Quitar los corchetes del cuerpo y la palabra "return" — el return está implícito.
(a) => a + 100;

// 3. Suprimir los paréntesis de los argumentos
a => a + 100;
```

Por ejemplo, si la función tiene varios argumentos o no tiene argumentos, es posible usar los paréntesis alrededor de los argumentos:

```
// Función tradicional
function (a, b){
  return a + b + 100;
}

// Función flecha
(a, b) => a + b + 100;

// Función tradicional (sin argumentos)
let a = 4;
let b = 2;
function (){
  return a + b + 100;
}

// Función flecha (sin argumentos)
let a = 4;
let b = 2;
() => a + b + 100;
```

Del mismo modo, si el cuerpo requiere líneas de procesamiento adicionales, se deberá volver a introducir los corchetes y el "return":

```
// Función tradicional
function (a, b){
  let prueba = 42;
  return a + b + prueba;
}
```

```
// Función flecha
(a, b) => {
  let prueba = 42;
  return a + b + prueba;
}
```

En las funciones con nombre las expresiones de flecha se tratan como variables:

```
// Función tradicional
function prueba (a){
  return a + 100;
}

// Función flecha
let prueba = a => a + 100;
```

Resumen

En esta Unidad…

Trabajamos con la Introducción a JavaScript

En la próxima Unidad…

Trabajaremos con variables y operadores