3. Variables: var VS let y const

Las variables son uno de los conceptos fundamentales en cualquier lenguaje de programación, y JavaScript no es la excepción. En este capítulo, exploraremos qué son las variables, los tipos de datos que pueden almacenar y cómo declararlas correctamente utilizando `let`, `var` y `const`. Además, aprenderemos sobre los comentarios en el código, una herramienta esencial para documentar y organizar nuestro trabajo.

Tipos de Datos en JavaScript

JavaScript es un lenguaje **tipado dinámicamente**, lo que significa que no es necesario declarar explícitamente el tipo de dato de una variable al crearla. Sin embargo, cada valor en JavaScript pertenece a un tipo de dato específico. Estos se dividen en dos categorías principales: **primitivos** y **compuestos**.

Tipos de Datos Primitivos

Los tipos de datos primitivos son los bloques básicos de información en JavaScript. Son inmutables, lo que significa que no pueden modificarse directamente:

1. **String**:

   Representa texto y se define entre comillas simples (`'`) o dobles (`"`). También puede usarse la sintaxis de plantilla literal con backticks (`` ` ``) para incluir expresiones dinámicas.

   Ejemplo:

   ```
   let nombre = "Juan";
   let saludo = `Hola, ${nombre}`;
   ```

2. **Number**:

   Incluye tanto números enteros como decimales. JavaScript no distingue entre ellos.

   Ejemplo:

   ```
   let edad = 25;
   let precio = 19.99;
   ```

3. **Boolean**:

   Representa valores lógicos: `true` o `false`.

   Ejemplo:

   ```
   let esMayorDeEdad = true;
   ```

4. **Null**:

   Representa la ausencia intencionada de un valor. Es un valor asignado explícitamente.

   Ejemplo:

```
let usuario = null;
```

5. **Undefined**:

   Indica que una variable ha sido declarada pero no tiene un valor asignado.

   Ejemplo:

   ```
   let direccion;
   console.log(direccion); // undefined
   ```

   (Las constantes no pueden crearse sin asignarles un valor)

6. **NaN (Not-a-Number)**:

   Representa un valor numérico inválido, generalmente resultado de operaciones matemáticas incorrectas.

   Ejemplo:

   ```
   let resultado = "hola" * 2; // NaN
   ```

Tipos de Datos Compuestos

Los tipos de datos compuestos son estructuras más complejas que pueden contener múltiples valores o incluso otras estructuras:

1. **Object**:

   Representa una colección de pares clave-valor. Es la base para muchas estructuras en JavaScript, como arrays, funciones y clases.

   Ejemplo:

   ```
   let persona = { nombre: "Ana", edad: 30 };
   ```

2. **Array**:

   Una lista ordenada de valores, indexados numéricamente.

   Ejemplo:

   ```
   let frutas = ["manzana", "plátano", "uva"];
   ```

3. **Function**:

   Un bloque de código reutilizable que puede ejecutarse cuando sea necesario.

   Ejemplo:

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

4. **Class**:

Una plantilla para crear objetos con propiedades y métodos específicos.

Ejemplo:

```
class Persona {
  constructor(nombre) {
    this.nombre = nombre;
  }
}
```

Diferencia entre `let`, `var` y su Alcance en Bloques

En JavaScript, existen tres formas principales de declarar variables: `var`, `let` y `const`. Cada una tiene características distintas que afectan su comportamiento y alcance.

`var`: Declara Variables Globales o de Función

Antes de ES6 (2015), `var` era la única forma de declarar variables. Tiene un **alcance de función** o **global**, lo que significa que está disponible en toda la función donde se declara o en todo el archivo si no está dentro de una función.

Ejemplo:

```
function ejemploVar() {
  if (true) {
    var mensaje = "Hola desde var";
  }
  console.log(mensaje); // "Hola desde var"
}
ejemploVar();
```

Sin embargo, el uso de `var` puede llevar a errores difíciles de depurar debido a su comportamiento de **hoisting** (elevación), que permite acceder a la variable antes de su declaración.

`let`: Declara Variables con Alcance de Bloque

Introducido en ES6, `let` resuelve muchos de los problemas asociados con `var`. Tiene un **alcance de bloque**, lo que significa que solo está disponible dentro del bloque donde se declara (delimitado por `{}`).

Ejemplo:

```
function ejemploLet() {
  if (true) {
    let mensaje = "Hola desde let";
  }
  console.log(mensaje); // ReferenceError: mensaje is not defined
}
ejemploLet();
```

El uso de `let` es recomendable para la mayoría de los casos, ya que evita errores relacionados con el alcance.

`const`: Declara Constantes

`const` se utiliza para declarar variables cuyo valor no cambiará después de ser asignado. Al igual que `let`, tiene un **alcance de bloque**.

Ejemplo:

```
const PI = 3.1416;
PI = 3.14; // TypeError: Assignment to constant variable.
```

Es importante destacar que, aunque el valor de una constante no puede cambiar, si es un objeto o array, sus propiedades o elementos internos sí pueden modificarse.

Comentarios en el Código

Los comentarios son fragmentos de texto ignorados por el intérprete de JavaScript, utilizados para explicar el código o desactivar partes temporalmente. Existen dos tipos de comentarios:

1. **Comentarios de Una Línea**:

   Se escriben usando `//`. Todo lo que sigue después de `//` en esa línea será ignorado.

   Ejemplo:

   ```
   // Esto es un comentario de una línea
   let numero = 10; // Esta variable almacena un número
   ```

2. **Comentarios de Varias Líneas**:

   Se escriben entre `/*` y `*/`. Pueden abarcar múltiples líneas.

   Ejemplo:

   ```
   /*
   Este es un comentario
   que abarca varias líneas.
   */
   ```

Los comentarios son una práctica esencial para mejorar la legibilidad del código y facilitar su mantenimiento, especialmente cuando trabajas en equipo.
