### **Tipos de Datos Complejos**

Los tipos de datos complejos en JavaScript son aquellos que pueden contener múltiples valores o estructuras de datos más elaboradas. A diferencia de los tipos primitivos, estos tipos son objetos.

#### **1. Objetos (Objects)**

* **Creación de objetos:**
* JavaScript
* let persona = { nombre: "Ana", edad: 30, ciudad: "Madrid", hobbies: ["leer", "cocinar", "viajar"] };
* Use code [with caution.](file:///C:/faq#coding)
* **Acceso a propiedades:**
* JavaScript
* console.log(persona.nombre); // Imprime: Anaconsole.log(persona["edad"]); // Otra forma de acceder: 30
* Use code [with caution.](file:///C:/faq#coding)
* **Modificación de propiedades:**
* JavaScript
* persona.edad = 31; persona.hobbies.push("programar");
* Use code [with caution.](file:///C:/faq#coding)
* **Objetos anidados:**
* JavaScript
* let mascota = { nombre: "Max", tipo: "perro", dueño: persona // Referencia al objeto persona };
* Use code [with caution.](file:///C:/faq#coding)

#### **2. Arreglos (Arrays)**

* **Creación de arreglos:**
* JavaScript
* let colores = ["rojo", "verde", "azul"]; let numeros = [1, 2, 3, 4, 5]; let mixto = [true, "hola", 3.14];
* Use code [with caution.](file:///C:/faq#coding)
* **Acceso a elementos:**
* JavaScript
* console.log(colores[1]); // Imprime: verde
* Use code [with caution.](file:///C:/faq#coding)
* **Modificación de elementos:**
* JavaScript
* numeros[2] = 10;
* Use code [with caution.](file:///C:/faq#coding)
* **Métodos de arreglos:**
* JavaScript
* colores.push("amarillo"); // Agrega un elemento al finalconsole.log(colores.length); // Obtiene la longitud del arreglo
* Use code [with caution.](file:///C:/faq#coding)

#### **3. Funciones (Functions)**

* **Declaración de funciones:**
* JavaScript
* function saludar(nombre) { console.log("Hola, " + nombre + "!"); }
* Use code [with caution.](file:///C:/faq#coding)
* **Invocación de funciones:**
* JavaScript
* saludar("Juan");
* Use code [with caution.](file:///C:/faq#coding)
* **Funciones como valores:**
* JavaScript
* let miFuncion = saludar; miFuncion("Ana");
* Use code [with caution.](file:///C:/faq#coding)
* **Funciones anónimas:**
* JavaScript
* let saludar2 = function(nombre) { console.log("Saludos, " + nombre); };
* Use code [with caution.](file:///C:/faq#coding)

#### **4. Otros tipos complejos (menos comunes):**

* **Date:** Representa un momento en el tiempo.
* **RegExp:** Representa expresiones regulares para búsqueda y manipulación de texto.
* **Map, Set:** Estructuras de datos más avanzadas para almacenar pares clave-valor o valores únicos.

### **¿Por qué son importantes los tipos de datos complejos?**

* **Organización de datos:** Permiten agrupar datos relacionados de manera lógica.
* **Reutilización de código:** Las funciones permiten crear bloques de código reutilizables.
* **Manejo de datos complejos:** Los arreglos y objetos son ideales para trabajar con grandes conjuntos de datos y estructuras jerárquicas.

### **Ejemplo práctico:**

JavaScript

let libro = { titulo: "El señor de los anillos", autor: "J.R.R. Tolkien", paginas: 1178, capitulos: ["La Comunidad del Anillo", "Las Dos Torres", "El Retorno del Rey"], resumen: function() { console.log("Este libro es una épica fantasía..."); } }; libro.resumen(); console.log("El libro tiene", libro.paginas, "páginas.");

Use code [with caution.](file:///C:/faq#coding)

**En resumen:**

Los tipos de datos complejos son esenciales para crear aplicaciones JavaScript más sofisticadas y dinámicas. Al comprender cómo utilizar objetos, arreglos y funciones, podrás modelar datos de manera más efectiva y escribir código más limpio y reutilizable.