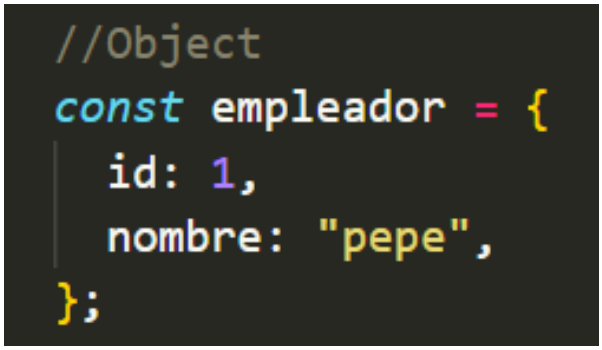
Tipos de datos compuestos

En JavaScript, los tipos de datos compuestos son aquellos que pueden contener múltiples valores o elementos, en contraste con los tipos de datos primitivos (como “number”, “string”, “boolean”, etc.). Los principales tipos de datos compuestos en JavaScript son:

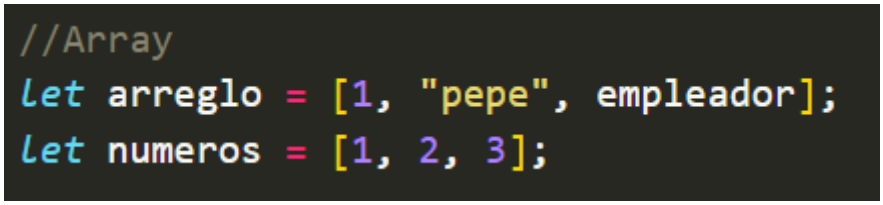
1. Object

- Un “Object” es una colección de pares clave-valor. Las claves (propiedades) son cadenas o símbolos, y los valores pueden ser de cualquier tipo, incluidos otros objetos, arrays, funciones, etc.



2. Array

- Un “Array” es una lista ordenada de elementos, que pueden ser de cualquier tipo. Los elementos están indexados numéricamente, comenzando desde 0.



3. Function

- Una “Function” es un bloque de código diseñado para realizar una tarea específica. En JavaScript, las funciones son objetos de primera clase, lo que significa que pueden ser asignadas a variables, pasadas como argumentos y devueltas por otras funciones.

```
//function
function saludar(nombre) {
  console.log("Hola, " + nombre);
}
```

4. Date

- El objeto “Date” se utiliza para trabajar con fechas y horas.

```
//Date
let ahora = new Date();
console.log(ahora); // M
```

5. RegExp (Expresiones Regulares)

- Una “RegExp” es un patrón utilizado para hacer coincidir cadenas de texto con una cierta estructura.

```
//Reg Exp
let email = "gabriel@gmail.com";
let expresionRegularEmail = /@/;

if (expresionRegularEmail.test(email)) {
  console.log("Es un email");
}
```

6. Map

- Un “Map” es una colección de pares clave-valor donde las claves pueden ser de cualquier tipo (no solo cadenas o símbolos).

```
//Map
let mapa = new Map();
mapa.set("clave1", "valor1");
mapa.set(2, "valor2");
```

7. Set

- Un “Set” es una colección de valores únicos. A diferencia de los arrays, no permite elementos duplicados.

```
let conjunto = new Set();
conjunto.add(1);
conjunto.add(2);
conjunto.add("Hola"); // No
```

8. WeakMap

- Similar a “Map”, pero las claves deben ser objetos, y las referencias a esas claves son "débiles", lo que permite que se eliminen de la memoria si no se referencian en otro lugar.

```
let weakMapa = new WeakMap();
let objeto = {};
weakMapa.set(objeto, "valor almacenado");

console.log(weakMapa.get(objeto));
```

9. WeakSet

- Similar a “Set”, pero solo acepta objetos como valores y, como en “WeakMap”, las referencias a los objetos son "débiles".

```
let weakConjunto = new WeakSet();
let objeto2 = {
  id: 123,
  nombre: "Gabriel",
};
weakConjunto.add(objeto2);
```