1: const obj = {};
Object.defineProperty(obj, "hidden",
{ value: "This is hidden",
enumerable: false, // Not enumerable
writable: true, // Optional: Can be modified
configurable: true // Optional: Can be deleted or changed });

1. Writable

Determines if the value of the property can be **changed**.

- **If `writable: true`**: The property value can be reassigned.
- **If `writable: false`**: The property value is read-only. Attempts to change it fail:
  - In **non-strict mode**, the operation silently fails.
  - In **strict mode**, it throws a `TypeError`.

2. Enumerable

Determines if the property will show up in **enumeration operations** like:

- `for...in`
- `Object.keys()`
- `Object.entries()`
- **If `enumerable: true`**: The property is visible during iteration.
- **If `enumerable: false`**: The property is hidden from enumeration methods but can still be accessed directly.

3. Configurable

Determines if the property's **descriptor itself can be modified** or if the property can be deleted.

- **If `configurable: true`**:
  - The property can be deleted using the `delete` operator.
  - The property's attributes (`writable`, `enumerable`, `configurable`) can be changed.
- **If `configurable: false`**:
  - The property cannot be deleted.
  - The property's attributes cannot be changed, except for `writable` (if it's `true`, it can be made `false`).

2: Object.getOwnPropertyDescriptor(obj, "key")
3: Obj.hasOwnProperty(key);
4: const obj = {};
Object.defineProperties(obj,
{ name: { value: 'Alice', writable: true, enumerable: true, configurable: true },
age: { value: 25, writable: true, enumerable: true, configurable: true } });

For in loop, use with object, don't use it with array….

Array.prototype.name = "Rohit"
arr.he = "Money"

For of loop