JavaScript Object Conversion Notes (toString, valueOf, ToPrimitive)

◇ Default Behavior

- Every JavaScript object has these methods:
  - `toString()` → returns `"[object Object]"`
  - `valueOf()` → returns the object itself

```
let obj = {};
console.log(obj.toString()); // "[object Object]"
console.log(obj.valueOf());  // {}
```

◇ Type Conversion Rule (IMPORTANT)

When an object is used with operators like `+`, `-`, `*`, `/`:

JavaScript internally calls **ToPrimitive**:

1. If hint is **number** (`-`, `*`, `/`)
   - First calls `valueOf()`
   - If not primitive → calls `toString()`
2. If hint is **string** (`String(obj)`, template literals)
   - First calls `toString()`
   - Then `valueOf()`

◇ Example 1: Overriding `toString()`

```
let obj = {
  toString() {
    return "90";
  },
  valueOf() {}
};

console.log(obj.toString()); // "90"
```

✔ `toString()` works because it returns a **primitive** (string)

◇ Example 2: Overriding `valueOf()`

```
let obj2 = {
  x: 10,
  valueOf() {
    return 10;
  }
};

console.log(obj2.valueOf()); // 10
```

✔ `valueOf()` returns a number → valid primitive

◇ Example 3: Default `valueOf()`

```
let obj3 = { x: 10 };

console.log(typeof obj3.valueOf()); // "object"
console.log(10 - obj3);             // NaN
```

✘ `valueOf()` returns object → not primitive
⇥ JS then calls `toString()` → "[object Object]"
⇥ `10 - "[object Object]"` → NaN

◇ Example 4: valueOf returns number (BEST CASE)

```
let obj4 = {
  x: 7,
  valueOf() {
    return 90;
  }
};

console.log(100 - obj4); // 10
```

✔ Steps:

- `obj4` → ToPrimitive (number hint)
- `valueOf()` → 90
- `100 - 90 = 10`

◇ Example 5: Invalid `toString()` return

```
let obj5 = {
  x: 7,
  toString() {
    return {};
  }
};

// console.log(100 - obj5); // ✖ TypeError
```

✖ `toString()` must return a **primitive**
✖ Returning object causes **TypeError**

◇ Example 6: `toString()` returns numeric string

```
let obj6 = {
  x: 8,
  toString() {
    return "88";
  }
};

console.log(100 - obj6); // 12
```

✔ Steps:

- `valueOf()` → object (ignored)
- `toString()` → "88"
- `100 - "88"` → 12

🧠 Key Takeaways (Interview Ready)

✔ `valueOf()` should return **number** for math operations
✔ `toString()` should return **string**
✔ Both must return **primitive values**
✔ `-` operator always prefers **number conversion**

★ One-Line Summary

> JavaScript converts objects using **ToPrimitive**, calling `valueOf()` first for math operations and `toString()` if needed.