Type Coercion in JavaScript

Type coercion is the automatic conversion of values from one data type to another by JavaScript when performing operations. It occurs in three forms:

- **Implicit Coercion** (Automatic conversion)
- **Explicit Coercion** (Manual conversion using functions)
- **Abstract Operations** (How JavaScript internally handles coercion)

1️⃣ Implicit Type Coercion (Automatic)

JavaScript automatically converts values when an operation expects a certain type.

📌 String Coercion (Number → String)

Occurs when using `+` (concatenation) with a string.

```
console.log("5" + 2);   // "52"  (Number is converted to String)
console.log(5 + "2");   // "52"
console.log("Hello" + true); // "Hellotrue"
console.log(5 + {}); // "5[object Object]" (Object gets converted to a string)
console.log(5 + NaN); // NaN (Invalid number)
console.log(5 + undefined); // NaN (Invalid number)
console.log(5 + null); // 5 (null converted to 0)
```

📌 Number Coercion (String → Number)

Occurs when using mathematical operators (`-` , `*` , `/` , `%`) with a string.

```
console.log("5" - 2);   // 3  (String "5" converted to Number)
console.log("5" * 2);   // 10
console.log("10" / "2"); // 5
console.log("10" - "abc"); // NaN (Invalid number)
```

✅ **Falsy values**: `0` , `""` , `null` , `undefined` , `NaN` , `false`
✅ **Truthy values**: Everything else (including `{}` , `[]` , `"0"` , `"false"`)

2️⃣ Explicit Type Coercion (Manual)

You can manually convert types using built-in functions.

String Conversion

```
console.log(String(123));   // "123"
console.log(String(true));  // "true"
console.log(String(null));  // "null"
console.log(String(undefined)); // "undefined"
console.log(String([1, 2, 3])); // "1,2,3" (Array to string)
```

Number Conversion

```
console.log(Number("123"));  // 123
console.log(Number("12.34")); // 12.34
console.log(Number("abc"));  // NaN (Cannot be converted)
console.log(Number(true));   // 1
console.log(Number(false));  // 0
console.log(Number(null));   // 0
console.log(Number(undefined)); // NaN
console.log(Number(""));   // 0
console.log(Number("99 88"));   // NaN
```

Boolean Conversion

```
console.log(Boolean(1));    // true
console.log(Boolean(0));    // false
console.log(Boolean(""));   // false
console.log(Boolean("abc")); // true
console.log(Boolean([]));   // true (Empty array is truthy)
console.log(Boolean({}));   // true (Empty object is truthy)
console.log(Boolean(null));     // false
console.log(Boolean(undefined)); // false
console.log(Boolean(NaN));      // false
```

3️⃣ Abstract Operations (How JavaScript Coerces Types Internally)

JavaScript follows three internal operations for type coercion:

1️⃣ ToPrimitive (Object to Primitive)

When an object is used in a string/number context, JavaScript calls `valueOf()` or `toString()` .

```
let obj = {
    valueOf: () => 42,
    toString: () => "Hello"
};

console.log(obj + 2); // "Hello2" (String context → Calls toString())
console.log(obj * 2); // 84 (Number context → Calls valueOf())
```

2️⃣ ToString (When converting to a string)

```
console.log(String(123));  // "123"
console.log(String([1, 2, 3])); // "1,2,3"
console.log(String({ a: 1 }));  // "[object Object]"
```

3️⃣ ToNumber (When converting to a number)

```
console.log(Number("123"));  // 123
console.log(Number("abc"));  // NaN
console.log(Number(""));     // 0
console.log(Number(null));   // 0
console.log(Number(undefined)); // NaN
```

🔴 Common Type Coercion Pitfalls

1️⃣ Unexpected String Concatenation

```
console.log(5 + "5");  // "55" (String Coercion)
console.log(5 - "5");  // 0 (Number Coercion)
```

2️⃣ Falsy Values in Conditionals

```
if ("0") {
    console.log("Truthy"); // Runs! (Non-empty strings are truthy)
}

if (0) {
    console.log("Falsy"); // Doesn't run (0 is falsy)
}
```

3️⃣ == vs. === (Type Coercion in Comparisons)

```
console.log(5 == "5");  // true (Type coercion happens)
console.log(5 === "5"); // false (Strict comparison, no coercion)
console.log(null == undefined); // true (Both are falsy)
console.log(null === undefined); // false (Different types)
console.log([] == 0); // true ([] → "" → 0)
console.log("0" == 0); // true ("0" → 0)
```

✅ **Solution**: Always use `===` to avoid unintended type conversions.

🔷 Summary

- **Implicit Coercion**: Happens automatically in operations.
- **Explicit Coercion**: Use `String()` , `Number()` , `Boolean()` for safer conversion.
- **Use `===`** to avoid unintended type coercion in comparisons.