JavaScript Type Coercion

1. Implicit Type Coercion

This happens automatically when JavaScript converts data types to make the operation valid. It often occurs during comparisons or arithmetic operations.

```
console.log("5" + 1);     // "51" (number 1 converted to string)
console.log("5" - 1);     // 4    (string "5" converted to number)
console.log(1 + true);    // 2    (true converted to 1)
console.log(1 + null);    // 1    (null converted to 0)
console.log(1 + undefined); // NaN (undefined cannot be converted to number)
```

Note:

Implicit coercion can lead to confusing results. JavaScript tries to help by converting one type to another based on the context, but this may not always behave as expected.

2. Explicit Type Coercion

This occurs when a developer manually converts a value to a different type using functions or methods.

```
String(123);        // "123"
Number("456");      // 456
Boolean(0);         // false
Boolean("hello");   // true
parseInt("42px");   // 42
parseFloat("3.14"); // 3.14
```

Best Practices:

- Prefer explicit coercion when possible to avoid bugs.
- Use strict comparison operators (===, !==) to avoid implicit type coercion.
- Always validate and sanitize user input before type conversion.