Here are conceptual examples of **typecasting** (type conversion) in JavaScript, demonstrating both **implicit** and **explicit** conversions:

---

1. Implicit Typecasting (Coercion)

JavaScript automatically converts one type to another where needed.

String to Number (Arithmetic Operations)

```
        let result = "5" * 2; // Implicitly converts "5" to a number
        console.log(result); // Output: 10
        console.log(typeof result); // Output: "number"
```

Concatenation with Strings

```
        let value = 5 + "2"; // Converts 5 to a string and concatenates
        console.log(value); // Output: "52"
        console.log(typeof value); // Output: "string"
```

Boolean to Number

```
        let isActive = true;
        console.log(isActive + 1); // true is implicitly converted to 1,
Output: 2
        console.log(false + 1); // false is converted to 0, Output: 1
```

Null to Number

```
        let value = null + 5; // null is converted to 0
        console.log(value); // Output: 5
```

Undefined to Number

```
        let value = undefined + 5; // undefined is converted to NaN
        console.log(value); // Output: NaN
```

2. Explicit Typecasting

Explicitly convert types using built-in functions or operators.

String to Number

```
let str = "42";
let num = Number(str); // Converts string to number
console.log(num); // Output: 42
console.log(typeof num); // Output: "number"
```

Number to String

```
let num = 42;
let str = String(num); // Converts number to string
console.log(str); // Output: "42"
console.log(typeof str); // Output: "string"
```

Boolean to String

```
let isLoggedIn = true;
let str = String(isLoggedIn); // Converts boolean to string
console.log(str); // Output: "true"
console.log(typeof str); // Output: "string"
```

String to Boolean

```
let value = Boolean("Hello"); // Non-empty string becomes true
console.log(value); // Output: true
console.log(Boolean("")); // Empty string becomes false, Output: false
```

Number to Boolean

```
let value = Boolean(1); // Non-zero number becomes true
console.log(value); // Output: true
console.log(Boolean(0)); // Zero becomes false, Output: false
```

Using parseInt() and parseFloat()

- Extract numbers from strings.

```
let str = "123.45abc";
console.log(parseInt(str)); // Output: 123 (integer part only)
console.log(parseFloat(str)); // Output: 123.45 (includes decimals
```

---

3. Special Cases

NaN (Not a Number)

- Any invalid arithmetic operation produces NaN.

```
let value = Number("abc"); // Cannot convert "abc" to a number
console.log(value); // Output: NaN
console.log(typeof value); // Output: "number"
```

Typecasting with the Unary + Operator

- Convert to a number.

```
let str = "42";
console.log(+str); // Output: 42
console.log(typeof +str); // Output: "number"
```

Double Negation for Boolean Conversion

- Use !! to explicitly convert to boolean.

```
console.log(!!"hello"); // Output: true
console.log(!!0); // Output: false
```

Converting Objects to Primitives

- Use toString() or valueOf().

```
let obj = { name: "John" };
console.log(obj.toString()); // Output: "[object Object]"
console.log(typeof obj.toString()); // Output: "string"
```

4. Examples of Mixed-Type Operations

Adding Numbers and Strings

```
let value = 5 + "5"; // Number 5 is converted to a string
console.log(value); // Output: "55"
```

Subtracting Strings

```
let value = "10" - "3"; // Strings are converted to numbers
console.log(value); // Output: 7
```

Comparison Operators

```
        console.log("5" == 5); // Implicitly converts "5" to a number, Output:
true
        console.log("5" === 5); // No type conversion, Output: false
```

5. Edge Cases

Empty String to Number

```
console.log(Number("")); // Output: 0
console.log(+"" === 0); // Output: true
```

Boolean and String

```
        console.log(true + "test"); // Converts true to "true", Output:
"truetest"
```

Object to String

```
    let obj = { name: "Alice" };
       console.log(String(obj)); // Output: "[object Object]"
```

Let me know if you'd like further clarification or practical challenges!