

# TypeScript Basics for Automation Testers – Day 5

**Topic:** Operators in TypeScript (Part 1 – Arithmetic, Assignment, and Relational Operators)

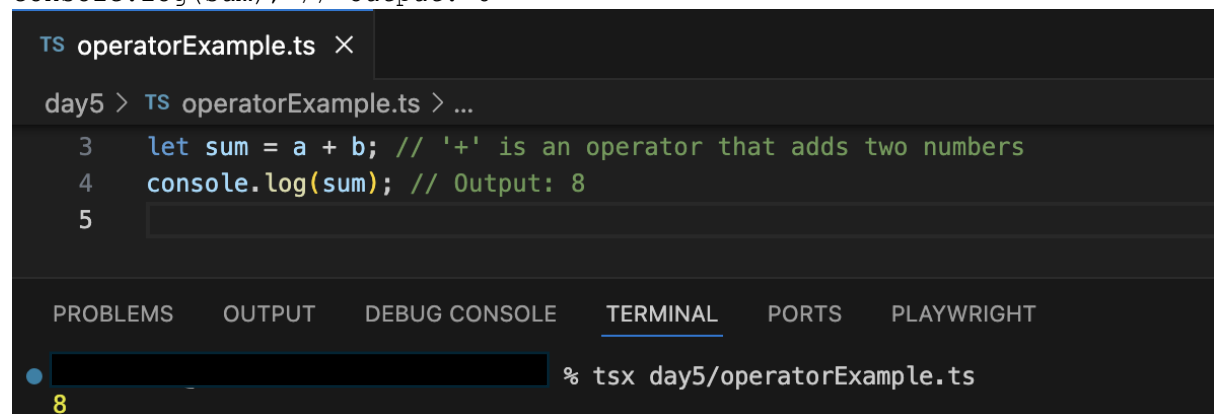
---

## What is an Operator?

The word **operator** means *someone or something that performs an operation or action*. In programming, **operators** are **symbols used to perform operations on variables and values**.

**Example:**

```
let a: number = 5;
let b: number = 3;
let sum = a + b; // '+' is an operator that adds two numbers
console.log(sum); // Output: 8
```



The screenshot shows a VS Code editor window with a file named `operatorExample.ts`. The code inside is:

```
3 let sum = a + b; // '+' is an operator that adds two numbers
4 console.log(sum); // Output: 8
5
```

The terminal at the bottom shows the command `% tsx day5/operatorExample.ts` being executed, and the output is `8`.

## Types of Operators in TypeScript

There are six main types of operators:

1. Arithmetic Operators
  2. Assignment Operators
  3. Relational (Comparison) Operators
  4. Logical Operators
  5. Increment and Decrement Operators
  6. Ternary (Conditional) Operator
-

# 1. Arithmetic Operators

## Meaning of “Arithmetic”:

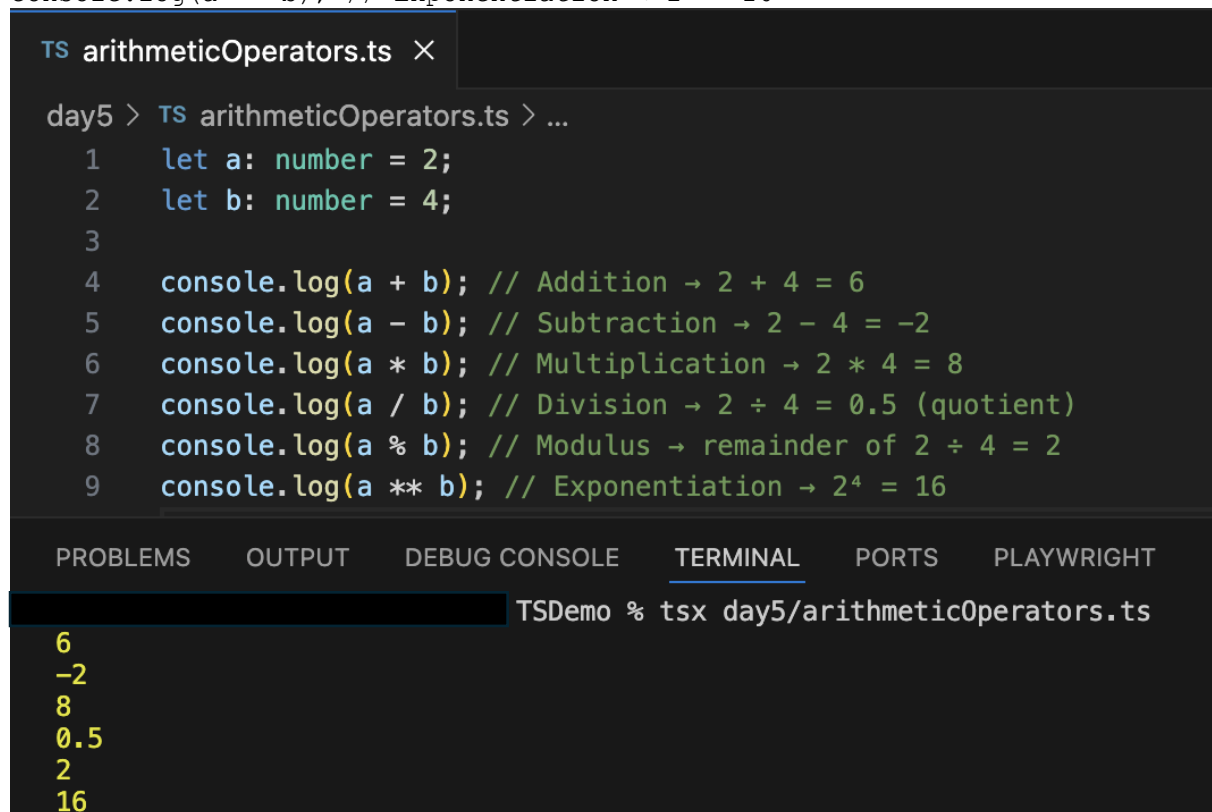
*arithmetic* means the branch of mathematics dealing with numbers and basic operations like addition, subtraction, multiplication, and division.

## In TypeScript:

Arithmetic operators are used to perform basic mathematical operations.

```
let a: number = 2;
let b: number = 4;

console.log(a + b); // Addition → 2 + 4 = 6
console.log(a - b); // Subtraction → 2 - 4 = -2
console.log(a * b); // Multiplication → 2 * 4 = 8
console.log(a / b); // Division → 2 ÷ 4 = 0.5 (quotient)
console.log(a % b); // Modulus → remainder of 2 ÷ 4 = 2
console.log(a ** b); // Exponentiation → 24 = 16
```



The screenshot shows a VS Code editor window with a file named 'arithmeticOperators.ts'. The code in the editor is as follows:

```
1 let a: number = 2;
2 let b: number = 4;
3
4 console.log(a + b); // Addition → 2 + 4 = 6
5 console.log(a - b); // Subtraction → 2 - 4 = -2
6 console.log(a * b); // Multiplication → 2 * 4 = 8
7 console.log(a / b); // Division → 2 ÷ 4 = 0.5 (quotient)
8 console.log(a % b); // Modulus → remainder of 2 ÷ 4 = 2
9 console.log(a ** b); // Exponentiation → 24 = 16
```

The terminal output at the bottom shows the results of these operations:

```
6
-2
8
0.5
2
16
```

## Note: Division vs Modulus

- **Division (/)** → gives **quotient** (how many times the number fits).  
Example:  $10 / 3 = 3.33$  → quotient = 3
- **Modulus (%)** → gives **remainder** after division.  
Example:  $10 \% 3 = 1$  → remainder = 1

Tip:

Division = how many times it goes inside

Modulus = what's left over after division

---

## 2. Assignment Operators

### Meaning of “Assignment”:

*assign* means **to give or allocate something to someone.**

In programming, **assignment operators** assign values to variables.

### Basic assignment:

```
let a: number = 10;
let b: number = 5;
a = b; // assigns value of b to a
console.log(a); // Output: 5
```

---

### Short-hand Assignment Operators

We can combine arithmetic and assignment together for simplicity. These are called **short-hand operators**.

```
let a: number = 10;
let b: number = 5;

a += b; // same as a = a + b → a = 15
console.log(a);

a -= b; // same as a = a - b → a = 10
console.log(a);

a *= b; // same as a = a * b → a = 50
console.log(a);

a /= b; // same as a = a / b → a = 10
console.log(a);

a %= b; // same as a = a % b → a = 0
console.log(a);
```

```
TS assignmentOperators.ts ×
day5 > TS assignmentOperators.ts > ...
1   let a: number = 10;
2   let b: number = 5;
3
4   a += b; // same as a = a + b → a = 15
5   console.log(a);
6
7   a -= b; // same as a = a - b → a = 10
8   console.log(a);
9
10  a *= b; // same as a = a * b → a = 50
11  console.log(a);
12
13  a /= b; // same as a = a / b → a = 10
14  console.log(a);
15
16  a %= b; // same as a = a % b → a = 0
17  console.log(a);

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  PLAYWRIGHT
% tsx day5/assignmentOperators.ts
15
10
50
10
0
```

## Important Difference Between = and ==

- **= (Single Equal)** → *Assignment operator* (used to assign value)
- **== (Double Equal)** → *Comparison operator* (used to compare two values)

```
let x = 10;
let y = 20;
```

```
x = y; // assigns y's value to x
console.log(x); // 20
```

```
console.log(x == y); // true → compares values only
```

```
TS equalVsDoubleEquals.ts ×
day5 > TS equalVsDoubleEquals.ts > ...
1  let x = 10;
2  let y = 20;
3
4  x = y; // assigns y's value to x
5  console.log(x); // 20
6
7  console.log(x == y); // true → compares values only

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  PLAYWRIGHT

TSDemo % tsx day5/equalVsDoubleEquals.ts
20
true
```

### 3.Relational or Comparison Operators

#### Meaning of “Relational”:

The word *relational* means **showing or describing a connection or comparison between two things**.

#### In TypeScript:

These operators are used to **compare two values** and return a **Boolean result** (`true` or `false`).

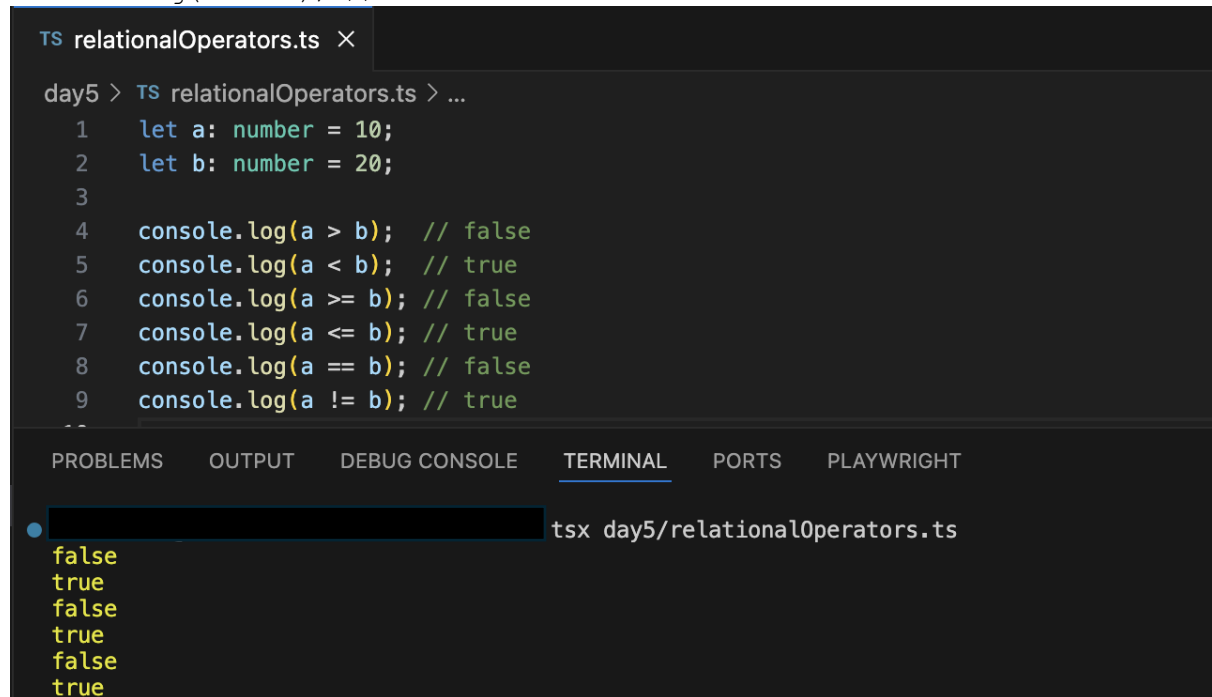
#### Common Relational Operators

Operator	Description	Example	Result
>	Greater than	10 > 5	true
<	Less than	10 < 5	false
>=	Greater than or equal to	10 >= 10	true
<=	Less than or equal to	10 <= 5	false
==	Equality (compares only values)	10 == "10"	true
!=	Not equal to	10 != 5	true
===	Strict equality (compares value + type)	10 === "10"	false

## Example:

```
let a: number = 10;
let b: number = 20;

console.log(a > b); // false
console.log(a < b); // true
console.log(a >= b); // false
console.log(a <= b); // true
console.log(a == b); // false
console.log(a !== b); // true
```



The screenshot shows a VS Code editor window with a file named `relationalOperators.ts`. The code in the file is as follows:

```
1 let a: number = 10;
2 let b: number = 20;
3
4 console.log(a > b); // false
5 console.log(a < b); // true
6 console.log(a >= b); // false
7 console.log(a <= b); // true
8 console.log(a == b); // false
9 console.log(a !== b); // true
```

The terminal at the bottom shows the command `tsx day5/relationalOperators.ts` being executed, with the following output:

```
false
true
false
true
false
true
```

## Double Equal (==) vs Triple Equal (===)

```
let num1: number = 10;
let num2: any = "10";

console.log(num1 == num2); // true → compares only values
console.log(num1 === num2); // false → compares both value & type
```

### Explanation:

- `==` checks only the **value**, not the **type**.
- `===` checks both **value** and **datatype** — hence safer and more reliable.

```
TS equalityComparison.ts ×
day5 > TS equalityComparison.ts > [?] num2
1   let num1: number = 10;
2   let num2: any = "10";
3   ⚡
4   console.log(num1 == num2); // true → compares only values
5   console.log(num1 === num2); // false → compares both value & type
6
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  PLAYWRIGHT
● TSDemo % tsx day5/equalityComparison.ts
true
false
```

---

## Questions

1. What is an operator in programming?
2. How many types of operators are there in TypeScript?
3. What is the difference between division (/) and modulus (%)?
4. Why are assignment operators called short-hand operators?
5. What is the main use of relational operators?
6. What is the difference between = and ==?
7. What is the difference between == and ===?
8. What kind of value do comparison operators return?

---

## Answers

1. An operator is a symbol used to perform an operation (like +, -, =, >) on variables or values.
  2. There are six types: Arithmetic, Assignment, Relational, Logical, Increment/Decrement, and Ternary.
  3. Division (/) gives quotient; Modulus (%) gives remainder.
  4. They are short-hand because they combine arithmetic and assignment in one step (e.g., a += b).
  5. Relational operators compare two values and return true or false.
  6. = assigns a value; == compares two values.
  7. == checks only value; === checks both value and data type.
  8. They always return a Boolean value — true or false.
-