

TypeScript Basics for Automation Testers – Day 5

Topic: Operators in TypeScript (Part 2 – Logical, Increment & Decrement, Ternary / Conditional Operators)

Difference Between Operators and Operands

Operator:

An **operator** is a symbol that tells the compiler or interpreter to perform a specific operation (like addition, subtraction, comparison, etc.). It defines **what action** needs to be done.

Operand:

An **operand** is the **value or variable** on which the operator acts. It defines **what is being operated on**.

Example:

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

- Here, **+** is the **operator** (it performs addition).
- **a** and **b** are **operands** (the values being added).
- The result **c** stores the outcome of the operation.

If **operator** is the **action**, then **operand** is the **object** of that action.

Like in “2 + 3”,

+ → operator

2 and **3** → operands

Logical Operators

What is “logical”?

logical means dealing with true/false reasoning or combining conditions to get a boolean result.

How it is related here: Logical operators combine boolean expressions and return a boolean (true / false). Common logical operators: **&&** (AND), **||** (OR), **!** (NOT).

Truth behaviour (short):

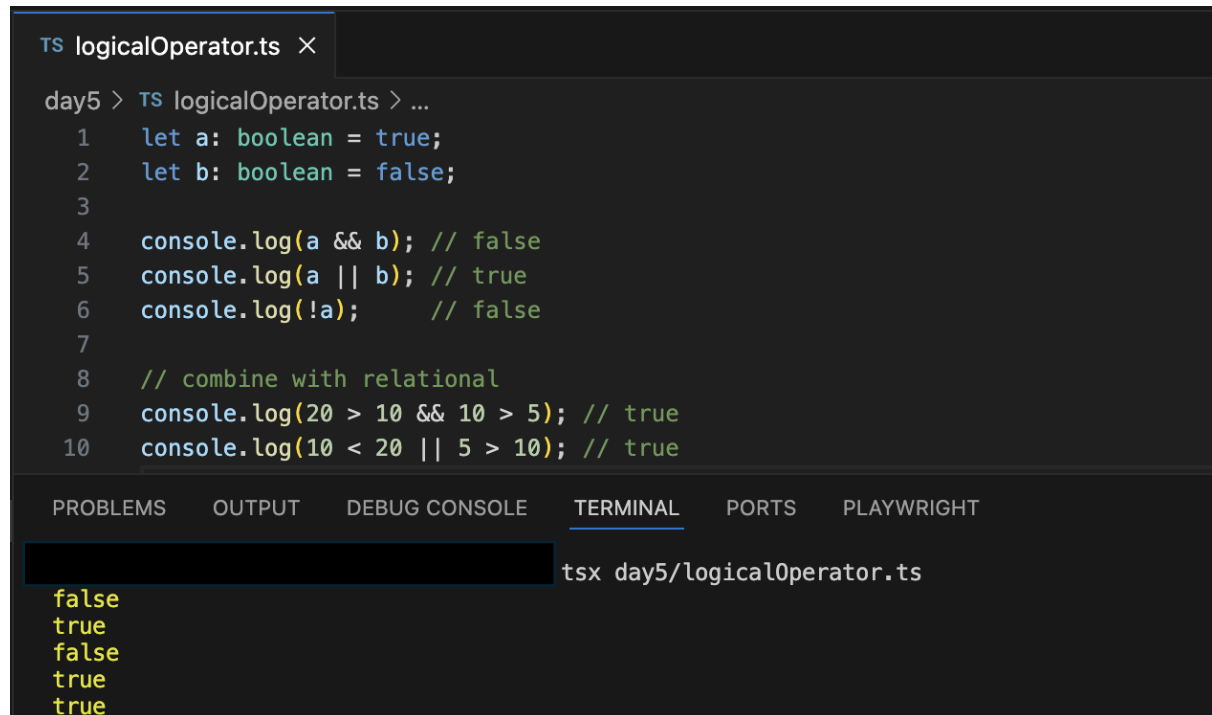
- **&&** → true only if both operands are true.
- **||** → true if any one operand is true.
- **!** → flips boolean value.

Example

```
let a: boolean = true;
let b: boolean = false;

console.log(a && b); // false
console.log(a || b); // true
console.log(!a);     // false

// combine with relational
console.log(20 > 10 && 10 > 5); // true
console.log(10 < 20 || 5 > 10); // true
```



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

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

Below the editor, the **TERMINAL** tab is active, showing the command `tsx day5/logicalOperator.ts` and its output:

```
false
true
false
true
true
```

Increment and Decrement Operators

What is it: Operators to increase (`++`) or decrease (`--`) a numeric value by one.

Post vs Pre behaviour:

- **Post-increment (`x++`):** returns the old value, then increases the variable.
 - **Pre-increment (`++x`):** increases first, then returns the new value.
- Same for `--` (decrement).

Example

```
let x: number = 10;

let postResult: number = x++; // post increment: postResult = 10, x becomes 11
console.log("postResult:", postResult); // 10
console.log("x after post++:", x);      // 11

x = 10; // reset
let preResult: number = ++x; // pre increment: x becomes 11, preResult = 11
console.log("preResult:", preResult); // 11
console.log("x after pre++:", x);     // 11

// Post-decrement
x = 10;
let postDec = x--; // postDec = 10, x becomes 9
console.log("postDec:", postDec); // 10
console.log("x after post--:", x); // 9

// Pre-decrement
x = 10;
let preDec = --x; // x becomes 9, preDec = 9
console.log("preDec:", preDec); // 9
console.log("x after pre--:", x); // 9
```

TS incrementDecrement.ts X

day5 > TS incrementDecrement.ts > ...

```
1  let x: number = 10;
2
3  let postResult: number = x++; // post increment: postResult = 10, x becomes 11
4  console.log("postResult:", postResult); // 10
5  console.log("x after post++:", x);      // 11
6
7  x = 10; // reset
8  let preResult: number = ++x; // pre increment: x becomes 11, preResult = 11
9  console.log("preResult:", preResult); // 11
10 console.log("x after pre++:", x);     // 11
11
12 // Post-decrement
13 x = 10;
14 let postDec = x--; // postDec = 10, x becomes 9
15 console.log("postDec:", postDec); // 10
16 console.log("x after post--:", x); // 9
17
18 // Pre-decrement
19 x = 10;
20 let preDec = --x; // x becomes 9, preDec = 9
21 console.log("preDec:", preDec); // 9
22 console.log("x after pre--:", x); // 9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS PLAYWRIGHT

tsx day5/incrementDecrement.ts

```
postResult: 10
x after post++: 11
preResult: 11
x after pre++: 11
postDec: 10
x after post--: 9
preDec: 9
x after pre--: 9
```

Ternary / Conditional Operator

What is it: A compact conditional expression that returns one of two values based on a boolean condition. It's the inline form of `if-else`.

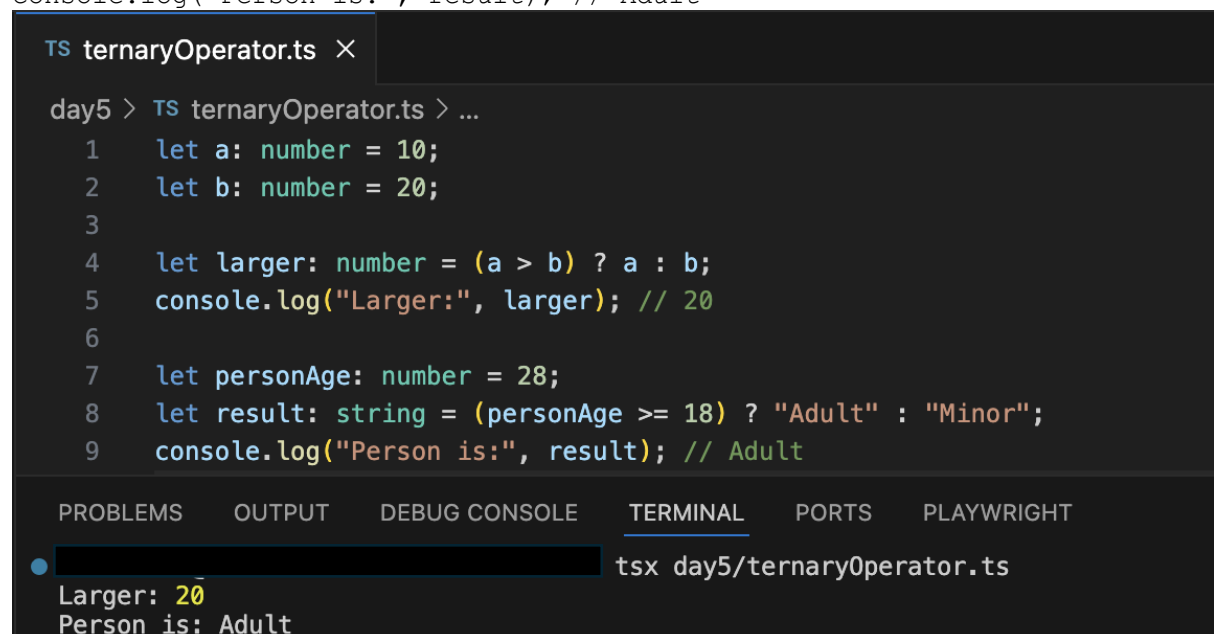
Syntax: `condition ? valueIfTrue : valueIfFalse`

Example

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

let larger: number = (a > b) ? a : b;
console.log("Larger:", larger); // 20

let personAge: number = 28;
let result: string = (personAge >= 18) ? "Adult" : "Minor";
console.log("Person is:", result); // Adult
```



The screenshot shows a code editor with a file named `ternaryOperator.ts`. The code is as follows:

```
1 let a: number = 10;
2 let b: number = 20;
3
4 let larger: number = (a > b) ? a : b;
5 console.log("Larger:", larger); // 20
6
7 let personAge: number = 28;
8 let result: string = (personAge >= 18) ? "Adult" : "Minor";
9 console.log("Person is:", result); // Adult
```

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

```
Larger: 20
Person is: Adult
```

Questions

1. What does a logical operator do?
2. Name the three logical operators and their basic behavior.
3. What value type do logical operators return?
4. What is the difference between post-increment (`x++`) and pre-increment (`++x`)?
5. What will `let r = x++ + ++x;` do conceptually?
6. What is the ternary operator and what is its basic syntax?
7. When would you prefer a ternary operator over `if-else`?
8. Which of the following returns true: `(true && false) || !false`?

Answers

A1. A logical operator combines boolean expressions and returns a boolean result (`true` or `false`).

A2. `&&` (AND) — true if both operands are true.

`||` (OR) — true if at least one operand is true.

`!` (NOT) — inverts the boolean value.

A3. Logical operators return a **boolean** value (`true` or `false`).

A4. Post-increment (`x++`) returns the variable's current value then increments it. Pre-increment (`++x`) increments first, then returns the new value.

A5. `let r = x++ + ++x;` mixes post and pre increments: first `x++` yields old `x`, then `++x` increments and yields new `x`. The expression result depends on initial `x` and can be confusing — test it in code to see exact numeric result. (Practice to observe behavior.)

A6. The ternary operator is `condition ? valueIfTrue : valueIfFalse`.

A7. Use ternary for short simple conditional assignments when it improves readability; use `if-else` for multi-line or complex logic.

A8. Evaluate: `(true && false) || !false → (false) || true → true`. So the answer is **true**.
