Operators in TypeScript (as in JavaScript) are special symbols or keywords that are used to perform operations on operands. These operations can be arithmetic, logical, bitwise, relational, etc. Here's an overview of the different types of operators available in TypeScript:

These operators help to perform various operations, from simple arithmetic to complex type assertions and manipulations, making TypeScript a powerful and flexible language for development.

**1. Arithmetic Operators**

Arithmetic operators are used to perform mathematical operations.

* + Addition
* - Subtraction
* \* Multiplication
* / Division
* % Modulus (remainder)
* ++ Increment
* -- Decrement

let a = 10;

let b = 5;

console.log(a + b); // 15

console.log(a - b); // 5

console.log(a \* b); // 50

console.log(a / b); // 2

console.log(a % b); // 0

a++;

console.log(a); // 11

b--;

console.log(b); // 4

**2. Assignment Operators**

Assignment operators are used to assign values to variables.

* = Assignment
* += Addition assignment
* -= Subtraction assignment
* \*= Multiplication assignment
* /= Division assignment
* %= Modulus assignment

let x = 10;

x += 5; // equivalent to x = x + 5

console.log(x); // 15

x -= 3; // equivalent to x = x - 3

console.log(x); // 12

x \*= 2; // equivalent to x = x \* 2

console.log(x); // 24

x /= 4; // equivalent to x = x / 4

console.log(x); // 6

x %= 2; // equivalent to x = x % 2

console.log(x); // 0

**3. Comparison Operators**

Comparison operators are used to compare two values.

* == Equal to
* === Strict equal to (value and type)
* != Not equal to
* !== Strict not equal to (value and type)
* > Greater than
* < Less than
* >= Greater than or equal to
* <= Less than or equal to

let p = 10;

let q = '10';

console.log(p == q); // true

console.log(p === q); // false

console.log(p != q); // false

console.log(p !== q); // true

console.log(p > 5); // true

console.log(p < 15); // true

console.log(p >= 10); // true

console.log(p <= 10); // true

**4. Logical Operators**

Logical operators are used to perform logical operations.

* && Logical AND
* || Logical OR
* ! Logical NOT

let isTrue = true;

let isFalse = false;

console.log(isTrue && isFalse); // false

console.log(isTrue || isFalse); // true

console.log(!isTrue); // false

console.log(!isFalse); // true

**5. Bitwise Operators**

Bitwise operators are used to perform operations on binary numbers.

* & AND
* | OR
* ^ XOR
* ~ NOT
* << Left shift
* >> Right shift
* >>> Zero-fill right shift

let r = 5; // 0101 in binary

let s = 3; // 0011 in binary

console.log(r & s); // 1 (0001 in binary)

console.log(r | s); // 7 (0111 in binary)

console.log(r ^ s); // 6 (0110 in binary)

console.log(~r); // -6 (inverts all bits)

console.log(r << 1); // 10 (1010 in binary)

console.log(r >> 1); // 2 (0010 in binary)

console.log(r >>> 1); // 2 (0010 in binary, zero-fill)

### Conditional (Ternary) Operator

The conditional operator is a shorthand for an if-else statement.

* ? : Ternary operator

let age = 18;

let canVote = (age >= 18) ? "Yes" : "No";

console.log(canVote); // Yes

### Type Operators

TypeScript also includes operators that are specific to type-related operations.

* typeof Type query
* instanceof Instance query
* as Type assertion

let str = "Hello, world!";

console.log(typeof str); // string

class Person {}

let person = new Person();

console.log(person instanceof Person); // true

let someValue: any = "Hello, TypeScript";

let strLength: number = (someValue as string).length;

console.log(strLength); // 15

### 8. Other Operators

* in Property in an object
* new Create an instance of an object
* delete Delete a property from an object

let obj = { name: "Alice", age: 25 };

console.log("name" in obj); // true

let date = new Date();

console.log(date instanceof Date); // true

delete obj.age;

console.log("age" in obj); // false