**Types of Operators in JavaScript**

JavaScript has a variety of operators that are used to perform operations on variables and values. Here’s a comprehensive list of the different types of operators available in JavaScript:

**1. Arithmetic Operators**

These operators are used to perform arithmetic on numbers.

**Addition (`+`):** Adds two numbers.

```javascript

let x = 5 + 2; // 7

```

**Subtraction (`-`):** Subtracts one number from another.

```javascript

let x = 5 - 2; // 3

```

**Multiplication (`\*`):** Multiplies two numbers.

```javascript

let x = 5 \* 2; // 10

```

**Division (`/`):** Divides one number by another.

```javascript

let x = 5 / 2; // 2.5

```

**Modulus (`%`):** Returns the division remainder.

```javascript

let x = 5 % 2; // 1

```

**Exponentiation (`\*\*`):** Raises the first operand to the power of the second operand.

```javascript

let x = 5 \*\* 2; // 25

```

**Increment (`++`):** Increases a variable by one.

```javascript

let x = 5;

x++; // x is now 6

```

**Decrement (`--`):** Decreases a variable by one.

```javascript

let x = 5;

x--; // x is now 4

```

**2. Assignment Operators**

These operators are used to assign values to variables.

**Assignment (`=`):** Assigns a value to a variable.

```javascript

let x = 5;

```

**Addition assignment (`+=`):** Adds a value and assigns the result to a variable.

```javascript

let x = 5;

x += 2; // x is now 7

```

**Subtraction assignment (`-=`):** Subtracts a value and assigns the result to a variable.

```javascript

let x = 5;

x -= 2; // x is now 3

```

**Multiplication assignment (`\*=`):** Multiplies by a value and assigns the result to a variable.

```javascript

let x = 5;

x \*= 2; // x is now 10

```

**Division assignment (`/=`):** Divides by a value and assigns the result to a variable.

```javascript

let x = 5;

x /= 2; // x is now 2.5

```

**Modulus assignment (`%=`):** Takes the modulus and assigns the result to a variable.

```javascript

let x = 5;

x %= 2; // x is now 1

```

**Exponentiation assignment (`\*\*=`):** Raises to a power and assigns the result to a variable.

```javascript

let x = 5;

x \*\*= 2; // x is now 25

```

**3. Comparison Operators**

These operators compare two values and return a boolean value.

**Equal to (`==`):** Compares two values for equality.

```javascript

5 == 5; // true

```

**Not equal to (`!=`):** Compares two values for inequality.

```javascript

5 != 3; // true

```

**Strict equal to (`===`):** Compares two values for equality and type.

```javascript

5 === 5; // true

5 === '5'; // false

```

**Strict not equal to (`!==`):** Compares two values for inequality and type.

```javascript

5 !== '5'; // true

```

**Greater than (`>`):** Checks if the left value is greater than the right value.

```javascript

5 > 3; // true

```

**Greater than or equal to (`>=`):** Checks if the left value is greater than or equal to the right value.

```javascript

5 >= 5; // true

```

**Less than (`<`):** Checks if the left value is less than the right value.

```javascript

3 < 5; // true

```

**Less than or equal to (`<=`):** Checks if the left value is less than or equal to the right value.

```javascript

3 <= 5; // true

```

**4. Logical Operators**

These operators are used to perform logical operations.

**Logical AND (`&&`):** Returns true if both operands are true.

```javascript

true && true; // true

```

**Logical OR (`||`):** Returns true if at least one of the operands is true.

```javascript

true || false; // true

```

**Logical NOT (`!`):** Returns the inverse boolean value.

```javascript

!true; // false

```

**5. Bitwise Operators**

These operators are used to perform bitwise operations on binary numbers.

**AND (`&`):** Performs a bitwise AND.

```javascript

5 & 1; // 1 (0101 & 0001)

```

**OR (`|`):** Performs a bitwise OR.

```javascript

5 | 1; // 5 (0101 | 0001)

```

**NOT (`~`):** Performs a bitwise NOT.

```javascript

~5; // -6 (inverts the bits)

```

**XOR (`^`):** Performs a bitwise XOR.

```javascript

5 ^ 1; // 4 (0101 ^ 0001)

```

**Left shift (`<<`):** Shifts bits to the left.

```javascript

5 << 1; // 10 (0101 << 1)

```

**Right shift (`>>`):** Shifts bits to the right.

```javascript

5 >> 1; // 2 (0101 >> 1)

```

**Zero fill right shift (`>>>`):** Shifts bits to the right with zero fill.

```javascript

5 >>> 1; // 2 (0101 >>> 1)

```

**6. String Operators**

These operators are used to manipulate strings.

**Concatenation (`+`):** Concatenates two strings.

```javascript

'Hello ' + 'World'; // 'Hello World'

```

**Concatenation assignment (`+=`):** Adds and assigns a string to a variable.

```javascript

let str = 'Hello';

str += ' World'; // 'Hello World'

```

**7. Conditional (Ternary) Operator**

This operator is used to assign a value based on a condition.

**Ternary (`? :`):** Assigns a value based on a condition.

```javascript

let x = 5;

let result = (x > 0) ? 'Positive' : 'Negative';

// result is 'Positive'

```

**8. Type Operators**

These operators are used to determine or change the type of a variable.

**typeof:** Returns the type of a variable.

```javascript

typeof 5; // 'number'

```

**instanceof:** Checks if an object is an instance of a particular class.

```javascript

let arr = [];

arr instanceof Array; // true

```

**void:** Evaluates an expression and returns undefined.

```javascript

void 0; // undefined

```

**9. Comma Operator**

This operator is used to evaluate multiple expressions.

**Comma (`,`):** Evaluates multiple expressions and returns the last one.

```javascript

let a = (1, 2, 3); // a is 3

```

**10. Spread and Rest Operators**

These operators are used for working with arrays and function arguments.

**Spread (`...`):** Expands an array or object.

```javascript

let arr = [1, 2, 3];

let newArr = [...arr, 4, 5]; // [1, 2, 3, 4, 5]

```

**Rest (`...`):** Condenses multiple elements into an array.

```javascript

function sum(...args) {

return args.reduce((acc, val) => acc + val, 0);

}

sum(1, 2, 3); // 6

```

These operators provide a powerful toolkit for manipulating data and controlling the flow of a JavaScript program.