##### JavaScript Operators

JavaScript operators are special symbols that perform operations on one or more operands (values).

## JavaScript Operator Types:

1. **Arithmetic Operators**
2. **Assignment Operators**
3. **Comparison Operators**
4. **Logical Operators**
5. **Bitwise Operators**
6. **String Operators**
7. **Miscellaneous Operators**

## JavaScript Arithmetic Operators

We use arithmetic operators to perform ****arithmetic calculations**** like addition, subtraction, etc. For example,

5 - 3; // 2

Here, we used the - operator to subtract ****3**** from ****5****.

### **Commonly Used Arithmetic Operators**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| **+** | **Addition** | **3 + 4 // 7** |
| **-** | **Subtraction** | **5 - 3 // 2** |
| **\*** | **Multiplication** | **2 \* 3 // 6** |
| **/** | **Division** | **4 / 2 // 2** |
| **%** | **Remainder** | **5 % 2 // 1** |
| **++** | **Increment (increments by **1**)** | **++5 or 5++ // 6** |
| **--** | **Decrement (decrements by **1**)** | **--4 or 4-- // 3** |
| **\*\*** | **Exponentiation (Power)** | **4 \*\* 2 // 16** |

### **Example 1: Arithmetic Operators in JavaScript**

**let x = 5;**

**// addition operator**

**console.log("Addition: x + 3 = ", x + 3);**

**// subtraction operator**

**console.log("Subtraction: x - 3 =", x - 3);**

**// multiplication operator**

**console.log("Multiplication: x \* 3 =", x \* 3);**

**// division operator**

**console.log("Division: x / 3 =", x / 3);**

**// remainder operator**

**console.log("Remainder: x % 3 =", x % 3);**

**// increment operator**

**console.log("Increment: ++x =", ++x);**

**// decrement operator**

**console.log("Decrement: --x =", --x);**

**// exponentiation operator**

**console.log("Exponentiation: x \*\* 3 =", x \*\* 3);**

****Output****

**Addition: x + 3 = 8**

**Subtraction: x - 3 = 2**

**Multiplication: x \* 3 = 15**

**Division: x / 3 = 1.6666666666666667**

**Remainder: x % 3 = 2**

**Increment: ++x = 6**

**Decrement: --x = 5**

**Exponentiation: x \*\* 3 = 125**

****Note:**** The increment operator ++ adds ****1**** to the operand. And, the decrement operator -- decreases the value of the operand by ****1****.

## 2. JavaScript Assignment Operators

We use assignment operators to ****assign**** values to variables. For example,

let x = 5;

Here, we used the = operator to assign the value ****5**** to the variable x.

### **Commonly Used Assignment Operators**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| **=** | **Assignment Operator** | **a = 7;** |
| **+=** | **Addition Assignment** | **a += 5; // a = a + 5** |
| **-=** | **Subtraction Assignment** | **a -= 2; // a = a - 2** |
| **\*=** | **Multiplication Assignment** | **a \*= 3; // a = a \* 3** |
| **/=** | **Division Assignment** | **a /= 2; // a = a / 2** |
| **%=** | **Remainder Assignment** | **a %= 2; // a = a % 2** |
| **\*\*=** | **Exponentiation Assignment** | **a \*\*= 2; // a = a\*\*2** |

### **Example 2: Assignment Operators in JavaScript**

**// assignment operator**

**let a = 7;**

**console.log("Assignment: a = 7, a =", a);**

**// addition assignment operator**

**a += 5; // a = a + 5**

**console.log("Addition Assignment: a += 5, a =", a);**

**// subtraction assignment operator**

**a -= 5; // a = a - 5**

**console.log("Subtraction Assignment: a -= 5, a =", a);**

**// multiplication assignment operator**

**a \*= 2; // a = a \* 2**

**console.log("Multiplication Assignment: a \*= 2, a =", a);**

**// division assignment operator**

**a /= 2; // a = a / 2**

**console.log("Division Assignment: a /= 2, a =", a);**

**// remainder assignment operator**

**a %= 2; // a = a % 2**

**console.log("Remainder Assignment: a %= 2, a =", a);**

**// exponentiation assignment operator**

**a \*\*= 2; // a = a\*\*2**

**console.log("Exponentiation Assignment: a \*\*= 7, a =", a);**

****Output****

**Assignment: a = 7, a = 7**

**Addition Assignment: a += 5, a = 12**

**Subtraction Assignment: a -= 5, a = 7**

**Multiplication Assignment: a \*= 2, a = 14**

**Division Assignment: a /= 2, a = 7**

**Remainder Assignment: a %= 2, a = 1**

**Exponentiation Assignment: a \*\*= 7, a = 1**

## 3. JavaScript Comparison Operators

We use comparison operators to ****compare**** two values and return a [boolean](https://www.programiz.com/javascript/booleans) value (true or false). For example,

**const a = 3, b = 2;**

**console.log(a > b);**

**// Output: true**

Here, we have used the > comparison operator to check whether a (whose value is ****3****) is greater than b (whose value is ****2****).

Since ****3**** is greater than ****2****, we get true as output.

### **Commonly Used Comparison Operators**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Meaning** | **Example** |
| **==** | **Equal to** | **3 == 5 gives us false** |
| **!=** | **Not equal to** | **3 != 4 gives us true** |
| **>** | **Greater than** | **4 > 4 gives us false** |
| **<** | **Less than** | **3 < 3 gives us false** |
| **>=** | **Greater than or equal to** | **4 >= 4 gives us true** |
| **<=** | **Less than or equal to** | **3 <= 3 gives us true** |
| **===** | **Strictly equal to** | **3 === "3" gives us false** |
| **!==** | **Strictly not equal to** | **3 !== "3" gives us true** |

### **Example 3: Comparison Operators in JavaScript**

**// equal to operator**

**console.log("Equal to: 2 == 2 is", 2 == 2);**

**// not equal operator**

**console.log("Not equal to: 3 != 3 is", 3 != 3);**

**// strictly equal to operator**

**console.log("Strictly equal to: 2 === '2' is", 2 === '2');**

**// strictly not equal to operator**

**console.log("Strictly not equal to: 2 !== '2' is", 2 !== '2');**

**// greater than operator**

**console.log("Greater than: 3 > 3 is", 3 > 3);**

**// less than operator**

**console.log("Less than: 2 > 2 is", 2 > 2);**

**// greater than or equal to operator**

**console.log("Greater than or equal to: 3 >= 3 is", 3 >= 3);**

**// less than or equal to operator**

**console.log("Less than or equal to: 2 <= 2 is", 2 <= 2);**

****Output****

**Equal to: 2 == 2 is true**

**Not equal to: 3 != 3 is false**

**Strictly equal to: 2 === '2' is false**

**Strictly not equal to: 2 !== '2' is true**

**Greater than: 3 > 3 is false**

**Less than: 2 > 2 is false**

**Greater than or equal to: 3 >= 3 is true**

**Less than or equal to: 2 <= 2 is true**

Difference between equality (== and !=) and strict equality (=== and !==) operators.

The equality operators (== and !=) [convert](https://www.programiz.com/javascript/type-conversion) both operands to the same type before comparing their values. For example,

console.log(3 == "3"); // true

Here, we used the == operator to compare the number ****3**** and the string 3.

By default, JavaScript converts string 3 to number ****3**** and compares the values.

However, the strict equality operators (=== and !==) do not convert operand types before comparing their values. For example,

console.log(4 === "4"); // false

Here, JavaScript didn't convert string 4 to number ****4**** before comparing their values.

Thus, the result is false, as number ****4**** isn't equal to string 4.

## 4. JavaScript Logical Operators

We use logical operators to perform logical operations on boolean expressions. For example,

**const x = 5, y = 3;**

**console.log((x < 6) && (y < 5));**

**// Output: true**

Here, && is the logical operator ****AND****. Since both x < 6 and y < 5 are true, the combined result is true.

### **Commonly Used Logical Operators**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Syntax** | **Description** |
| **&& (Logical AND)** | **expression1 && expression2** | **true only if both expression1 and expression2 are true** |
| **|| (Logical OR)** | **expression1 || expression2** | **true if either expression1 or expression2 is true** |
| **! (Logical NOT)** | **!expression** | **false if expression is true and vice versa** |

### **Example 4: Logical Operators in JavaScript**

**let x = 3;**

**// logical AND**

**console.log((x < 5) && (x > 0)); // true**

**console.log((x < 5) && (x > 6)); // false**

**// logical OR**

**console.log((x > 2) || (x > 5)); // true**

**console.log((x > 3) || (x < 0)); // false**

**// logical NOT**

**console.log(!(x == 3)); // false**

**console.log(!(x < 2)); // true**

****Note:**** We use [comparison and logical operators](https://www.programiz.com/javascript/comparison-logical) in decision-making and loops. You will learn about them in detail in later tutorials.

**5. JavaScript Bitwise Operators**

We use bitwise operators to perform ****binary operations**** on integers.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| **&** | **Bitwise AND** | **5 & 3 // 1** |
| **|** | **Bitwise OR** | **5 | 3 // 7** |
| **^** | **Bitwise XOR** | **5 ^ 3 // 6** |
| **~** | **Bitwise NOT** | **~5 // -6** |
| **<<** | **Left shift** | **5 << 1 // 10** |
| **>>** | **Sign-propagating right shift** | **-10 >> 1 // -5** |
| **>>>** | **Zero-fill right shift** | **-10 >>> 1 // 2147483643** |

****Note:**** We rarely use bitwise operators in everyday programming. If you are interested, visit [JavaScript Bitwise Operators](https://www.programiz.com/javascript/bitwise-operators) to learn more.

**6. JavaScript String Concatenation Operator**

In JavaScript, you can also use the + operator to concatenate (join) two strings. For example,

**let str1 = "Hel", str2 = "lo";**

**console.log(str1 + str2);**

**// Output: Hello**

Here, we used the + operator to concatenate str1 and str2.

**7. JavaScript Miscellaneous Operators**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| **,** | ****Comma**: Evaluates multiple operands and returns the value of the last operand.** | **let a = (1, 3, 4); // 4** |
| **?:** | ****Ternary**: Returns value based on the condition.** | **(50 > 40) ? "pass" : "fail"; // "pass"** |
| **typeof** | **Returns the data type of the variable.** | **typeof 3; // "number"** |
| **instanceof** | **Returns true if the specified object is a valid object of the specified class.** | **objectX instanceof ClassX** |
| **void** | **Discards any expression's return value.** | **void(x) // undefined** |

***….………..TQ……………***