Basic Operators in Js

Contents:

Operators

An operator is used to transform one or more values into a single resultant value.

The value to which the operator is applied is referred as operands.

A combination of an operator and its operands is referred to as an expression.

Js-Operators

JavaScript supports following operators:

- Arithmetic Operators
- Assingnment Operators
- Comparison Operators
- Ternary/Conditional Operators
- Logical Operators
- Bitwise Operators
-

Arithmetic Operators

. Arithmetic operators are the most familiar operators because they are used every day to solve common math calculations.

+	Addition\
-	Subtraction
-	Multiplication(*)
/	Division
%	Modulus(Remainder after division)
++	Increment operator
- -	Decrement operator

Addition:

Examples:

```
let x = 5;

let y = 2;

let z = x + y;
```

O/p + 7

Subtraction:

```
let x = 5;

let y = 2;

let z = x - y;
```

O?p = 3

JavaScript Operators

Name	Operators
Arithmetic	+ , - , * , /
Comparison	==, ===, >=, <= ,!= ,!==
Logical	|| , && , !
Ternary	condition?true block:false block
Assignment	= , +=, -=, *=, /=, ^= ,%=

Category	Operator	Name/Description	Example	Result
Arithmetic	+	Addition	3+2	5
	-	Subtraction	3-2	1
	*	Multiplication	3*2	6
	/	Division	10/5	2
	%	Modulus	10%5	0
	++	Increment and then return value	X=3; ++X	4
		Return value and then increment	X=3; X++	3
	--	Decrement and then return value	X=3; --X	2
		Return value and then decrement	X=3; X--	3
Logical	&&	Logical “and” evaluates to true when both operands are true	3>2 && 5>3	False
	||	Logical “or” evaluates to true when either operand is true	3>1 || 2>5	True
	!	Logical “not” evaluates to true if the operand is false	3!=2	True
Comparison	==	Equal	5==9	False
	!=	Not equal	6!=4	True
	<	Less than	3<2	False
	<=	Less than or equal	5<=2	False
	>	Greater than	4>3	True
	>=	Greater than or equal	4>=4	True
String	+	Concatenation(join two strings	“A”+”BC”	ABC

Multiplying:

```
let x = 5;

let y = 2;

let z = x * y;

o/p = 10;
```

Dividing:

```
let x = 5;

let y = 2;

let z = x / y;

o/p = 2.5
```

Remainder(Modulus):

```
let x = 5;

let y = 2;

let z = x % y;

o/p = 1;
```

Incrementing:

```
let x = 5;

x++;

let z = x;

o/p = 6;
```

Decrementing:

```
let x = 5;

x--;

let z = x;

o/p = 4;
```

Exponentiation **

```
alert( 2 ** 2 ); // 2² = 4

alert( 2 ** 3 ); // 2³ = 8

alert( 2 ** 4 ); // 2⁴ = 16
```

Comparison Operators

. Comparison operators are used to compare two values.

==	Equal
!=	Not Equal
<=	Less than or equal to
>=	Greater than or equal to
===	Strictly equal
!==	Strictly not equal

. The equal(==) and not equal to (!=) operators perform type casting for equality

Ex: “5” == 5 ;

o/p: = true;

. The Strictly equal (===) and strictly not equal to (!==) operators perform type casting for equality.

Ex; “5” ===5

o/p: = false;

String Operators

. String Operators are those operators that are used to perform operations on strings.

. Currently, JavaScript supports only the String Concatenation (+) operator.

. This operator is used to join two strings.

Ex: “Far” + “ooq” o/p: Farooq

What are String Operators in JavaScript?

.JavaScript String Operators are used to manipulate and perform operations on strings. There are two operators which are used to modify strings in JavaScript. These operators help us to join one string to another string.

Type of JavaScript String Operators

.Concatenate Operator

.Concatenate Assignment Operator

String Concatenate Operator

Concatenate Operator in JavaScript combines strings using the ‘+’ operator and creates a new string that includes the contents of the original strings in which Concatenate string1 and string2, ensuring the first character of string2 immediately follows the last character of string1.

Ex: et str1 = "Farooq";

let str2 = "Shaik";

let result = (str1 + str2);

console.log(result);

o/p : **FarooqShaik**

String Concatenate Assignment Operator

In this, we perform a concatenation assignment by using the '+=' operator to add the value of a variable or string to an existing string variable.

Ex: let str1 = "Farooq";

let str2 = "ShaikG";

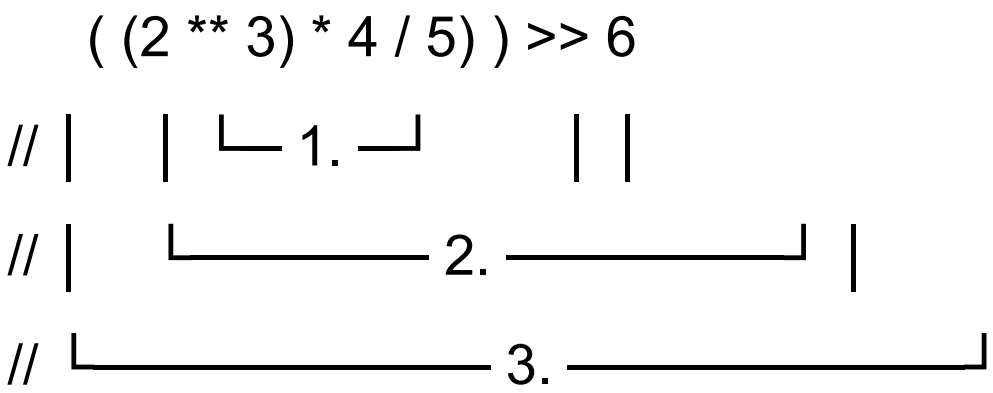
str1 += str2;

console.log(str1);

o/p : **FarooqShaoikG**

Operator Precedence

Operator precedence determines how operators are parsed concerning each other. Operators with higher precedence become the operands of operators with lower precedence.



Assignment Operator

JavaScript assignment operator is equal (=) which assigns the value of the right-hand operand to its left-hand operand. That is if a = b assigns the value of b to a.

```
x=10

y=20

x=y // Here, x is equal to 20

y=x // Here, y is equal to 10
```

Terinary Operator

JavaScript Ternary Operator (Conditional Operator) is a concise way to write a conditional (if-else) statement. Ternary Operator takes three operands i.e. condition, true value and false value. In this article, we are going to learn about Ternary Operator.

Input: let result = (10 > 0) ? true : false;

Output: true

Input: let message = (20 > 15) ? "Yes" : "No";

Output: Yes

Logical Operator

Logical operators are used to determine the logic between variables or values.

There are four logical operators in JavaScript: || (OR), && (AND), ! (NOT), ?? (Nullish Coalescing). Here we cover the first three, the ?? operator is in the next article.

Although they are called “logical”, they can be applied to values of any type, not only boolean. Their result can also be of any type.

|| (OR)

The “OR” operator is represented with two vertical line symbols:

result = a || b;

```
alert( true || true );   // true

alert( false || true );  // true

alert( true || false );  // true

alert( false || false ); // false
```

&& (AND)

The AND operator is represented with two ampersands &&:

result = a && b;

```
alert( true && true );   // true

alert( false && true );  // false

alert( true && false );  // false

alert( false && false ); // false
```

! (NOT)

The boolean NOT operator is represented with an exclamation sign !.

The syntax is pretty simple:

result = !value;

alert(!true); // **false**

alert(!o); // **true**

BitWise operator

Bitwise operators treat arguments as 32-bit integer numbers and work on the level of their binary representation.

These operators are not JavaScript-specific. They are supported in most programming languages.

The list of operators:

AND (&)

OR (|)

XOR (^)

NOT (~)

LEFT SHIFT (<<)

RIGHT SHIFT (>>)

ZERO-FILL RIGHT SHIFT (>>>)

These operators are used very rarely, when we need to fiddle with numbers on the very lowest (bitwise) level. We won't need these operators any time soon, as web development has little use of them, but in some special areas, such as cryptography, they are useful. You can read the Bitwise Operators chapter on MDN when a need arises.

Thank You