JavaScript

Module 2 - JavaScript Basics

Chapter 5 - Operators and Expressions

Aitrich

What are Operators

JavaScript operators are symbols that are used to perform operations on operands.

For example:

```
var sum=10+20;
```

Here, + is the arithmetic operator and = is the assignment operator.

Types of Operators

There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Assignment Operators
3. Comparison (Relational) Operators
4. String Operators
5. Logical Operators
6. Bitwise Operators
7. Ternary Operator
8. Type Operators

Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on numbers.

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
**	Exponentiation (raises the first operand to the power of the second operand.)	5 ** 2=25
/	Division	20/10 = 2
%	Modulus (returns Division Remainder)	20%10 = 0 5%2=1
++	Increment	var a=10; a++; Now a = 11
--	Decrement	var a=10; a--; Now a = 9

Operators and Operands

The numbers (in an arithmetic operation) are called operands.

The operation (to be performed between the two operands) is defined by an operator.

Operand	Operator	Operand
100	+	50

Arithmetic Operators-Example

```
<!DOCTYPE html>
<html>
<body>

<p id="msg"></p>

<script>
let a = 3;
let x = (100 + 50) * a;
document.getElementById("msg").innerHTML = x;
</script>

</body>
</html>
```

Output

450

Assignment Operators

Assignment operators assign values to JavaScript variables.

Operator	Name	Description	Example	Same As	Example with value
=	Simple Assignment Operator	Assigns a value to a variable	x = y	x = y	10+10 = 20
+=	Addition Assignment Operator	Add and assign	x += y	x = x + y	var a=10; a+=20; Now a = 30 let text = "Hello"; text += " World"; Now, text="Hello World"
-=	Subtraction Assignment Operator	Subtract and assign	x -= y	x = x - y	var a=20; a-=10; Now a = 10
**=	Exponentiation Assignment Operator	raises a variable to the power of the operand	x **= y	x = x ** y	let x = 10; x **= 5; Now x=100000

Assignment Operators

*=	Multiplication Assignment Operator	Multiply and assign	x *= y	x = x * y	var a=10; a*=20; Now a = 200
/=	Division Assignment Operator	Divide and assign	x /= y	x = x / y	var a=10; a/=2; Now a = 5
%=	Remainder Assignment Operator	assigns a remainder to a variable	x %= y	x = x % y	var a=10; a%=2; Now a = 0

Comparison(Relational) Operators

The JavaScript comparison operator compares the two operands.

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal value and equal type)	10==20 = false
!=	Not equal to	10!=20 = true
!==	Not Identical(not equal value or not equal type)	20!==20 = false
?	ternary operator	

Comparison(Relational)Operators

>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

String Comparison

All the comparison operators above can also be used on strings and Numbers.
Note that strings are compared alphabetically

```
<!DOCTYPE html>
<html>
<body>
<p id="msg"></p>
<script>
let text1 = "A";
let text2 = "B";
let result1 = text1 < text2;
let text3 = "20";
let text4 = "5";
let result2 = text3 < text4;
document.getElementById("msg").innerHTML = "Is A less
than B? " + result1 +"<br>"+ "Is 20 less than 5? " + result2;
</script>
</body>
</html>
```

Output

Is A less than B? true
Is 20 less than 5? true

String Addition

The + and += can also be used to add (concatenate) strings:

Note: When used on strings, the + operator is called the concatenation operator.

```
<!DOCTYPE html>
<html>
<body>
<p id="msg"></p>
<script>
let text1 = "Aitrich";
let text2 = "Academy";
let text3 = text1 + " " + text2;
let text4 = "What a very ";
text4 += "nice day";
document.getElementById("msg").innerHTML =
ext3+"<br>"+text4;
</script>
</body>
</html>
```

Output

Aitrich Academy
What a very nice day

Aitrich

Adding Strings and Numbers

Adding two numbers, will return the sum, but adding a number and a string will return a string

Note: If you add a number and a string, the result will be a string!

```
<!DOCTYPE html>
<html>
<body>
<p id="msg"></p>
<script>
let x = 5 + 5;
let y = "5" + 5;
let z = "Hello" + 5;
document.getElementById("msg").innerHTML = x + "<br>" + y + "<br>" + z;
</script>
</body>
</html>
```

Output

10
55
Hello5

Logical Operators

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false
!	Logical Not	!(10==20) = true

Bitwise Operators

The bitwise operators perform bitwise operations on 32 bits numbers.

Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

Operator	Description	Example1	Same as	Result	Decimal	Example2
&	AND	5 & 1	0101 & 0001	0001	1	(10==20 & 20==33) = false
|	OR	5 | 1	0101 | 0001	0101	5	(10==20 | 20==33) = false
~	NOT	~ 5	~0101	1010	10	(~10) = -10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4	(10==20 ^ 20==33) = false
<<	left shift	5 << 1	0101 << 1	1010	10	(10<<2) = 40
>>	right shift	5 >> 1	0101 >> 1	0010	2	(10>>2) = 2
>>>	unsigned right shift (or, right shift with zero)	5 >>> 1	0101 >>> 1	0010	2	(10>>>2) = 2

Ternary/Conditional Operator

The “Question mark” or “conditional” operator in JavaScript is a ternary operator that has three operands. It is the simplified operator of if/else.

Syntax: variable name = (condition) ? value if true : value if false

- **condition:** Expression to be evaluated which returns a boolean value.
- **value if true:** Value to be executed if the condition results in a true state.
- **value if false:** Value to be executed if the condition results in a false state.

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

Output: true

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

Output: Yes
```

```
let marks = 95;
    let result = (marks < 40) ?
"Unsatisfactory" :
        (marks < 60) ? "Average" :
            (marks < 80) ? "Good" :
"Excellent";
//Output will be “Excellent”
```

```
    let PMarks = 40
    let result = (PMarks > 39) ?
        "Pass" : "Fail";
//Output will be “Pass”
```

Type Operators

Operator	Description
typeof	Returns the type of a variable/Object
instanceof	Returns true if an object is an instance of given object type

Type-of Operator

You can use the typeof operator to find the data type of a JavaScript variable.

```
typeof "John"                 // Returns "string"
typeof 3.14                   // Returns "number"
typeof NaN                    // Returns "number"
typeof false                  // Returns "boolean"
typeof [1,2,3,4]              // Returns "object"
typeof {name:'John', age:34}  // Returns "object"
typeof new Date()             // Returns "object"
typeof function () {}         // Returns "function"
typeof myCar                  // Returns "undefined" *
typeof null                   // Returns "object"
```

Javascript Expressions

An expression is a combination of values, variables, and operators, which computes to a value.

Example:-

```
5 * 10
```

Expressions can also contain variable values:

Example:-

```
x * 10
```

Expressions can also contain numbers and strings.

Example:-

```
"Aitrich" + " " + "Academy"
```

Aitrich

Javascript Statements

Programming instructions to be "executed" by a computer are called statements.

```
let x, y, z; // Statement 1
x = 5; // Statement 2
y = 6; // Statement 3
z = x + y; // Statement 4
```

Operator Precedence

- JavaScript operator precedence determines the order in which operations are performed within an expression containing multiple operators.

Precedence Levels:

Grouping: Parentheses () have the highest precedence, followed by square brackets [] and curly braces {}. They control the order of evaluation within the brackets.

Exponentiation (**): Exponentiation (raising a number to a power) has the next highest precedence.

Multiplication (*) & Division (/): Multiplication and division have the same precedence and are evaluated from left to right.

Operator Precedence

- Addition (+) & Subtraction (-): Addition and subtraction also have the same precedence and are evaluated from left to right.
- Comparison (==, !=, >, <, etc.): Comparison operators determine relationships between values and are evaluated from left to right.
- Logical AND (&&): The logical AND operator has lower precedence than comparison operators.
- Logical OR (||): The logical OR operator has even lower precedence than logical AND.
- Assignment (=, +=, -=, etc.): Assignment operators have the lowest precedence and are evaluated from right to left.

Thank You