JS Operators:

There are different types of JavaScript operators:

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- String Operators
- Logical Operators
- Ternary Operators
- Bitwise Operators
- Type Operators

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

Operator Precedence

Operator precedence describes the order in which operations are performed in an arithmetic expression.

Example

let x = 100 + 50 * 3;

Is the result of the example above the same as 150 * 3, or is it the same as 100 + 150?

Is the addition or the multiplication done first?

As in traditional school mathematics, the multiplication is done first.

Multiplication (`*`) and division (`/`) have higher **precedence** than addition (`+`) and subtraction (`-`).

And (as in school mathematics) the precedence can be changed by using parentheses.

When using parentheses, the operations inside the parentheses are computed first:

Example

let x = (100 + 50) * 3;

When many operations have the same precedence (like addition and subtraction or multiplication and division), they are computed from left to right:

1. JavaScript Arithmetic Operators

Arithmetic Operators are used to perform arithmetic on numbers:

let a = 3;let x = (100 + 50) * a;

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division

%	Modulus (Division Remainder)
++	Increment
--	Decrement

2. JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

The **Addition Assignment Operator** (`+=`) adds a value to a variable.

```
let x = 10;
```

```
x += 5;
```

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

The `+=` assignment operator can also be used to add (concatenate) strings:

Example

let text1 = "What a very ";

text1 += "nice day";

The result of text will be:

`What a very nice day`

Note

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

3. JavaScript Comparison Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

---

4. JavaScript String Comparison

All the comparison operators above can also be used on strings:

Example

let text1 = "A";

let text2 = "B";

let result = text1 < text2;

Note that strings are compared alphabetically:

You can use the `localeCompare` method to compare two strings in the current locale. Here's the syntax:

```
string1.localeCompare(string2)
```

`locaelCompare` returns:

- 1 if `string1` is greater (higher in the alphabetical order) than `string2`
- 1 if `string1` is smaller (lower in the alphabetical order) than `string2`
- 0 if `string1` and `string2` are equal in the alphabetical order

Here are some examples comparing two strings:

```
const string1 = "hello"
const string2 = "world"

const compareValue = string1.localeCompare(string2)
// -1
```

It gives `-1` because, in the English locale, **h** in hello comes before **w** in the world (w is further down in the alphabetical order than h)

Another example:

```
const string1 = "banana"
const string2 = "back"

const compareValue = string1.localeCompare(string2)
// 1
```

The comparison above gives `1` because, in the English locale, ba**n** in banana comes after ba**c** in back.

Adding Strings and Numbers

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

Example

let x = 5 + 5;

let y = "5" + 5;

let z = "Hello" + 5;

The result of *x*, *y*, and *z* will be:

`10`

`55`

`Hello5`

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

5. JavaScript Logical Operators

Operator	Description
&&	logical and
||	logical or
!	logical not

Logical Operators

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

Given that `x = 6` and `y = 3`, the table below explains the logical operators:

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x == 5 || y == 5) is false
!	not	!(x == y) is true

6. Conditional (Ternary) Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax

variablename = (*condition*) ? *value1*:*value2*

Example

let voteable = (age < 18) ? "Too young":"Old enough";

If the variable age is a value below 18, the value of the variable voteable will be "Too young", otherwise the value of voteable will be "Old enough".

Comparing Different Types

Comparing data of different types may give unexpected results.

When comparing a string with a number, JavaScript will convert the string to a number when doing the comparison. An empty string converts to 0. A non-numeric string converts to `NaN` which is always `false`.

Case	Value
2 < 12	true
2 < "12"	true
2 < "John"	false
2 > "John"	false
2 == "John"	false
"2" < "12"	false
"2" > "12"	true
"2" == "12"	false

When comparing two strings, "2" will be greater than "12", because (alphabetically) 1 is less than 2.

The Nullish Coalescing Operator (??)

The `??` operator returns the first argument if it is not **nullish** (`null` or `undefined`).

Otherwise it returns the second argument.

Example

let name = null;

let text = "missing";

let result = name ?? text;

console.log(`the name is ${result}`);

Ans: the name is missing.

The Optional Chaining Operator (?.)

The `?.` operator returns `undefined` if an object is `undefined` or `null` (instead of throwing an error).

Example

const car = {type:"Fiat", model:"500", color:"white"};

let name= car?.name;

console.log(`the name is ${name}`);

Ans: the name is undefined.