

# TECHNOLOGY



## Coding Bootcamp

# TECHNOLOGY



## JavaScript



## Operators



# Learning Objectives

By the end of this lesson, you will be able to:

- 👁 Define operators and list the types of operators for identifying their different functions
- 👁 Demonstrate the use of operators with syntaxes and examples for effective coding implementation.
- 👁 Identify operator precedence for writing correct and efficient expressions



## Introduction to Operators

# What Are Operators?

They are symbols or keywords that tell the JavaScript engine to perform specific operations on one or more operands.



1

They are fundamental building blocks in programming and are essential for performing a variety of tasks.

2

They can manipulate a value or operand.

3

They are used to perform specific mathematical, logical, or other types of operations on one or more operands.



# Uses of Operators

Operators are used to:

Perform some specific  
mathematical or  
logical computations



Compare the values and  
perform the  
arithmetic operations

## Types of Operators



# Types of Operators

JavaScript provides various operators to perform different types of operations.



- 1 Arithmetic operators
- 2 Assignment operators
- 3 Comparison operators
- 4 Equality operators
- 5 Ternary operators
- 6 Logical operators
- 7 Bitwise operators



# Arithmetic Operators

They are used to perform mathematical operations on numbers.

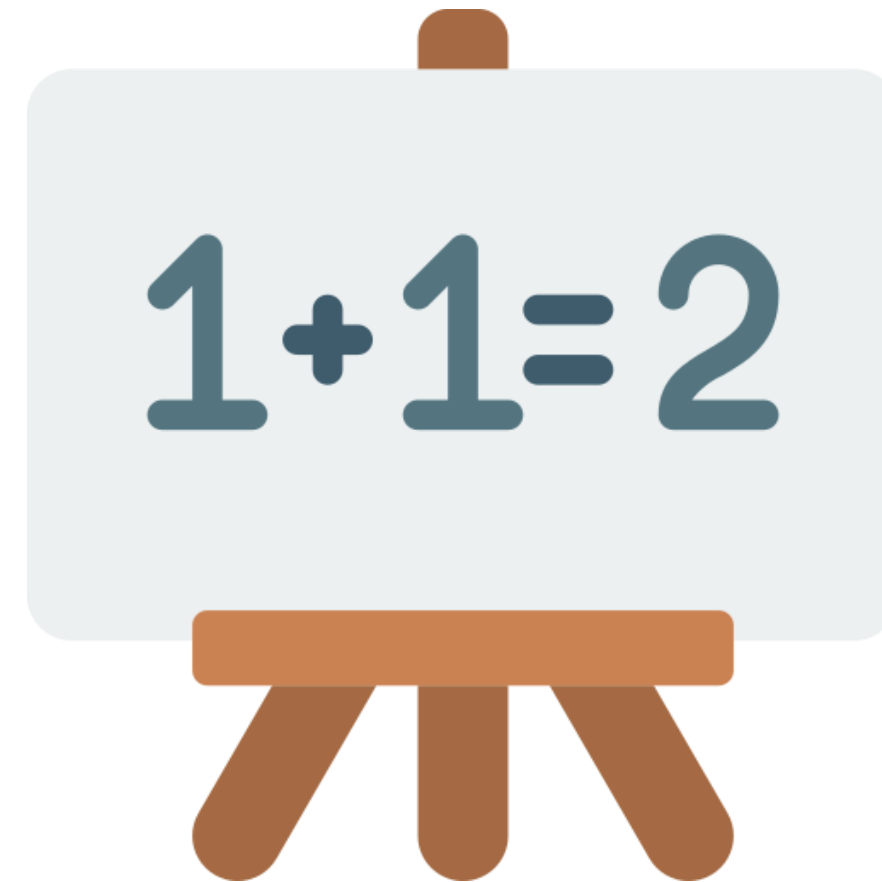
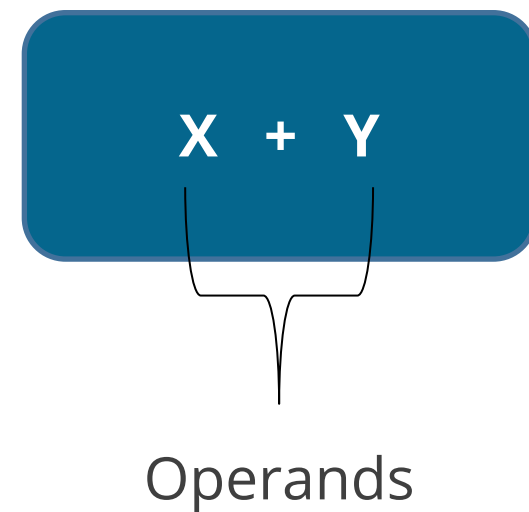
Arithmetic operators	Symbol
Addition	+
Subtraction	-
Division	/
Multiplication	*
Modulus	%
Increment	++
Decrement	--

They are essential for performing calculations and manipulating numerical data.



# Operands in Arithmetic Operations

Operands are the values or variables on which the operator performs the operation. In arithmetic operations, operands are the numbers that are manipulated by the arithmetic operators.

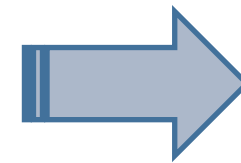




# Arithmetic Operators

## Example:

```
<!DOCTYPE html>
<html>
<body>
<h3>Arithmetic Operators</h3>
<p>Addition of two numbers</p>
<p id="test"></p>
<script>
let a = 40 +60;
Document.getElementById("test").innerHTML = a;
</script>
</html>
```



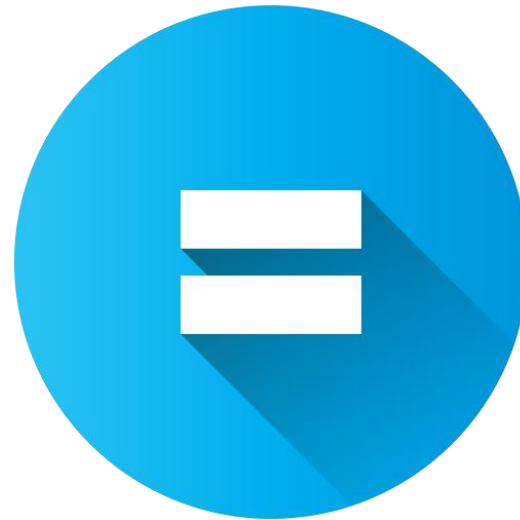
## Output

```
Arithmetic Operators
Addition of two numbers
100
```

# Assignment Operators

They are used to assign values to variables.

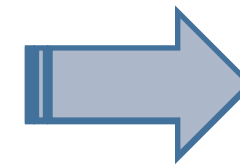
The assignment operator's symbol is (=).



# Assignment Operators

## Example:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Assignment Operators (=)</h2>
<p id="test"></p>
<script>
let a = 60;
document.getElementById("test").innerHTML = a;
</script>
</body>
</html>
```



## Output

```
JavaScript Assignment
Operators (=)
60
```



# Compound Assignment Operators

In addition to the basic assignment operator, there are compound assignment operators that perform an operation and assignment in one step. They are:

Assignment operators	Symbol
Addition assignment	<code>+=</code>
Subtraction assignment	<code>-=</code>
Multiplication assignment	<code>*=</code>
Division assignment	<code>/=</code>
Modulus assignment	<code>%=</code>



# Comparison Operators

They are used to compare two values and return a boolean result (true or false), essential for making decisions in code, such as conditional statements and loops.

Types of comparison operators are as follows:

Less than (<)

Greater than (>)

Less than or equal to  
(<=)


Greater than or equal  
to (>=)

Equal to (==)


Not equal to (!=)

# Comparison Operators


The list of comparison operators are:



**Less than (<):** Returns true if the value on the left side is less than the right-hand side value; otherwise, it returns the false value.




**Greater than (>):** Returns true if the value on the left side is greater than the value on the right; otherwise, it returns the false value.




**Less than or equal to (<=):** Returns true if the value on the left side is less than or equal to the right side; otherwise, it returns the false value.




# Comparison Operators



**Greater than or equal to ( $\geq$ ):** Returns true if the left side is greater than or equal to the value on the right; otherwise, it returns false.



**Equals to ( $===$ ):** Returns true if the value on the left side is equal to the value on the right side; else, it returns the false value.

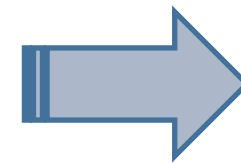


**Not equal to ( $!==$ ):** Returns true if the value on the left side is not equal to the value on the right; otherwise, it returns the false value.

# Comparison Operators

## Example:

```
<!DOCTYPE html>
<html>
<body>
<h3>Comparison example</h3>
<p>The comparison operators compare value and return
the Boolean value</p>
<script>
  let a = 10;
  let exp = a === 11;
  document.getElementById("test").innerHTML = exp;
</script>
</body>
</html>
```



## Output

```
Comparison example
The comparison operators
compare values and return the
Boolean value.
false
```

# Equality Operators

They are used to compare two expressions or values, determining whether the values are equivalent and returning a boolean result (true or false).

Types of equality operators are:

Equality operators	Symbol
Equal to	==
Strict equal to	===
Not equal	!=
Strict not equal to	!==

These operators are essential for making decisions in code that helps to compare values and execute different actions based on the results.

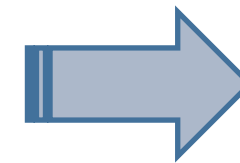




# Equality Operators

Example:

```
<!DOCTYPE html>
<html>
<body>
<p id="test"></p>
<script>
  var x = 10;
  document.getElementById("test").innerHTML = (x ==
10) ;
</script>
</body>
</html>
```

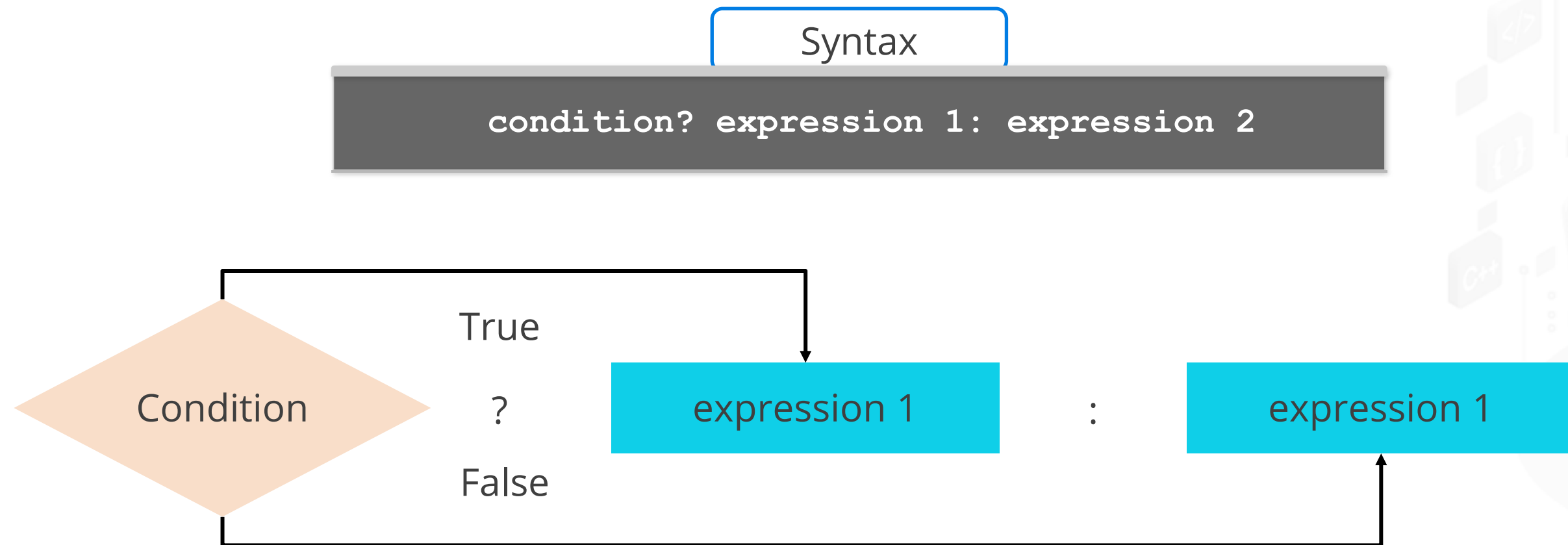


Output

true

# Ternary Operators

Ternary operators, also known as conditional operators, are used to evaluate a condition and return one of two values based on whether the condition is true or false



It takes three operands and evaluates a condition, returning one of two values depending on whether the condition is true or false.

# Logical Operators

They are used to perform logical operations, allowing you to combine multiple conditions or invert the result of a condition. There are two types of logical operators:

**&& (and)**

This operator will act as true if and only if the expressions on both sides are true.



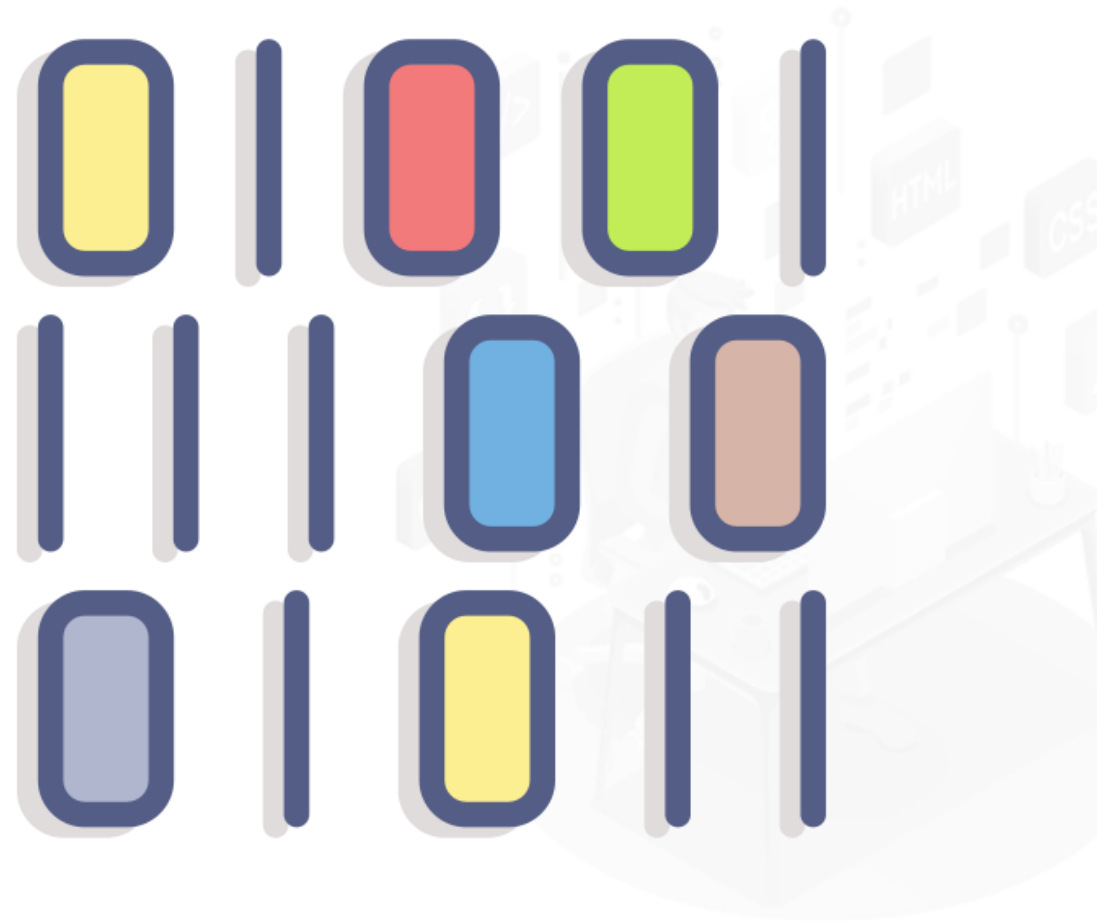
**|| (or)**

This operator will be true if the expression on either side of it is true; otherwise, its value is false.

# Bitwise Operator

They are used to perform bit-level operations on binary representations of integers.

Operators	Name	Example
&	Bitwise AND	$x \& y$
	Bitwise OR	$x   y$
~	Bitwise NOT	$\sim x$
<<	Left shift	$x \ll y$
>>	Right shift	$x \gg y$
>>>	Zero fill right shift	$x \ggg y$





# Operator Precedence

It determines the order in which operators are evaluated in an expression.  
Common precedence rules are:

Precedence level	Symbol
Highest precedence	Parentheses ( )
High precedence	Member access ( . ), function call ( ( ) ), array indexing ( [ ] )
Medium precedence	Multiplication ( * ), division ( / ), addition ( + ), subtraction ( - )
Low precedence	Comparison operators ( < , <= , > , >= ), equality operators ( == , != , === , !== )
Lowest precedence	Assignment operators ( = , += , -= )

Higher precedence operators become the operands of lower precedence operators.

# Differentiating between Equal to and Strict Equality Operator



## Problem Statement:

You have been asked to demonstrate the usage of the comparison operators `==` and `===` in JavaScript.

## Outcome:

By completing this task, you demonstrated the usage of the comparison operators `==` and `===` in JavaScript. You distinguished between `==`, which checks for equality after type coercion, and `===`, which checks for strict equality without type coercion.

**Note:** Refer to the demo document for detailed steps:  
[01\\_Differentiating\\_between\\_Equal\\_to\\_and\\_Strict\\_Equality\\_Operator](#)

# Assisted Practice: Guidelines

## Steps to be followed:

1. Distinguish between `==` and `===`



# Working with Logical Operators



## Problem Statement:

You have been asked to introduce logical operators in JavaScript and demonstrate their usage in making decisions based on multiple conditions.

## Outcome:

By completing this task, you introduced logical operators in JavaScript and demonstrated their usage in making decisions based on multiple conditions. You created a JavaScript file and implemented the logical AND, OR, and NOT operators to handle various conditional scenarios.

**Note:** Refer to the demo document for detailed steps:  
02\_Working\_with\_Logical\_Operators

# Assisted Practice: Guidelines

## Steps to be followed:

1. Create a JavaScript file
2. Implement logical AND operator
3. Implement logical OR operator
4. Implement logical NOT operator



## Key Takeaways

- Operators are symbols or keywords that tell the JavaScript engine to perform specific operations on one or more operands.
- The bitwise operators are used to perform bit-level operations on binary representations of integers.
- The comparison operators are used to compare two values and return a boolean result (true or false).
- The equality operator (==) compares two values or expressions, returning true or false based on their equivalence.
- Higher precedence operators become the operands of lower precedence operators.





# TECHNOLOGY

**Thank You**