**Day-5**

**Operators:**

Operators are symbols used to perform operations on variables and values

**1 ) Arithmetic Operators**: Used to perform arithmetic operations

let x = 10;

let y = 5;

let addition = x + y; // Addition

let subtraction = x - y; // Subtraction

let multiplication = x \* y; // Multiplication

let division = x / y; // Division

let expo=x\*\*y //Exponential

let modulus = x % y; // Modulus (remainder)

let increment = x++; // Increment

let decrement = y--; // Decrement

**2 ) Assignment Operators**: Used to assign values to variables.

let x = 10;

x += 5; // Equivalent to x = x + 5

x -= 5; // Equivalent to x = x - 5

x \*= 5; // Equivalent to x = x \* 5

x /= 5; // Equivalent to x = x / 5

**3 ) Comparison Operators**: Used to compare values. They return a boolean value - true or false.

let a = 10;

let b = 5;

console.log(a > b); // Greater than

console.log(a < b); // Less than

console.log(a >= b); // Greater than or equal to

console.log(a <= b); // Less than or equal to

console.log(a === b); // Equal to (strict equality)

console.log(a !== b); // Not equal to (strict inequality)

**4 ) Logical Operators**: Used to combine or manipulate boolean values.

let p = true;

let q = false;

console.log(p && q); // AND

console.log(p || q); // OR

console.log(!p); // NOT

**//logical and**

## **console.log(true && true); // true**

## **console.log(true && false); // false**

## **console.log(false && true); // false**

## **console.log(false && false); // false**

**//logical or**

## **console.log(true || true); // true**

## **console.log(true || false); // true**

## **console.log(false || true); // true**

## **console.log(false || false); // false**

**5 ) Ternary Operator (Conditional Operator):** Used to assign a value to a variable based on a condition.

## **condition ? expressionIfTrue : expressionIfFalse;**

let age = 20;

let status = age >= 18 ? "Adult" : "Minor";

console.log(status); // Output: 'Adult'

When nesting ternary operators, you place one ternary operation inside another. This is useful when you have multiple conditions to evaluate in a compact way. Here's an example of how a nested ternary operator works:

# **let result = condition1 ? value1 : condition2 ? value2 : value3;**

**Breakdown of Syntax:**

1. **First condition** (condition1):
   * If true, it will return value1.
   * If false, it will check the next condition (condition2).
2. **Second condition** (condition2):
   * If true, it will return value2.
   * If false, it will return value3.

**6 ) Nullish Coalescing Operator (??)**

It is a logical operator that returns its right-hand operand when its left-hand operand is **null** or **undefined**, and otherwise returns its left-hand operand. It's useful for providing default values in expressions without overriding valid falsy values like 0, NaN, or ''.

Syntax

let result = expression1 ?? expression2;

//case - 1

let name1 = null;

let defaultName1 = name1 ?? "yes it is null or undefined";

console.log(defaultName1); // Output: 'yes it is null or undefined'

//case - 2

let name2 = undefined;

let defaultName2 = name2 ?? "yes it is null or undefined";

console.log(defaultName2); // Output: 'yes it is null or undefined'

//case - 3

let name3 = "hello world";

let defaultName3 = name3 ?? "yes it is null or undefined";

console.log(defaultName3); // Output: hello world

Practical Applications

Handling Optional Function Parameters:

function hello(a){

var b=a ?? "Dear";

console.log("hello " + b);

}

hello();// we are not any value but still we are getting value because of Nullish Coalescing Operator

hello("teja")

**7 ) Optional Chaining Operator (?.)**

It is a powerful tool that allows for safe navigation through nested object properties, functions, and arrays. It prevents runtime errors that occur when accessing properties of null or undefined objects, returning undefined instead of throwing an error.

**Syntax**

let result = object?.property;

let result = object?.[property];

let result = object?.method?.();

// wihout using optional chaining operator

var obj = {

name: "tej",

state: {

name: "ap",

},

};

console.log(obj.obj.state); //it throws an error

// with using optional chaining operator

var obj = {

name: "tej",

state: {

name: "ap",

},

};

console.log(obj.obj?.state); //undefined instead of error

**8 ) Bitwise operator**

**What are bitwise operators?**

Bitwise operators are used to perform operations at the **bit** level, where a **bit** is the smallest unit of data in a computer, represented as either a **0** or a **1**. Every number in a computer is stored in binary (which is a series of 0s and 1s).

**AND (&)**

Compares two numbers **bit by bit**. If both bits are 1, the result is 1. Otherwise, it’s 0.

Think of it like a "both must be true" rule.

101 (binary for 5) & 011 (binary for 3) --------

001 (binary for 1)// Result: 1

* Sets each bit to 1 if both corresponding bits are 1.

**let a = 5; // 0101**

**let b = 3; // 0011**

**let result = a & b; // 0001 (1 in decimal)**

**OR (|)**

* Compares two numbers **bit by bit**. If **either** bit is 1, the result is 1. If both are 0, the result is 0.
* Think of it like "if either is true."

101 (binary for 5) | 011 (binary for 3) --------

111 (binary for 7)// Result: 7

* Sets each bit to 1 if at least one of the corresponding bits is 1.

**let a = 5; // 0101**

**let b = 3; // 0011**

**let result = a | b; // 0111 (7 in decimal)**

**Bitwise XOR (^)**

* Compares two numbers **bit by bit**. If the bits are different, the result is 1. If they are the same, the result is 0.
* Think of it like "one or the other, but not both."

101 (binary for 5) ^ 011 (binary for 3) --------

110 (binary for 6)// Result: 6

* Sets each bit to 1 if only one of the corresponding bits is 1.

**let a = 5; // 0101**

**let b = 3; // 0011**

**let result = a ^ b; // 0110 (6 in decimal)**

**9** **) String operators**

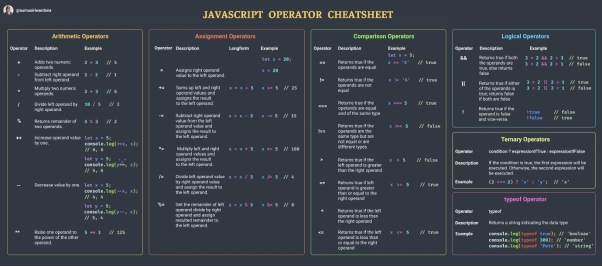
It is used to perform operations on string values. The primary string operator is the concatenation operator (+), but other operations involving strings include template literals, comparison operators, and methods available on string objects.

**Concatenation Operator (+)**

The concatenation operator combines two or more strings into a single string.

let greeting = "Hello, " + "world!";

console.log(greeting); // Outputs: "Hello, world!"



**Type Coercion**

Type coercion refers to the automatic or implicit conversion of values from one data type to another. This process happens in the background during operations involving values of different types.

**Types of Type Coercion**

**Implicit Coercion**: This occurs automatically when JavaScript encounters an operation involving different data types.

**Explicit Coercion**: This is when you manually convert a value from one type to another using functions or methods.

**Explicit Type Conversion**

JavaScript type conversion, allowing you to convert values from one data type to another.

1. **String()**: Converts a value to a string.

let num = 123;

let str = String(num);

console.log(str); // Output: "123"

1. **Number()**: Converts a value to a number.

let str = "123";

let num = Number(str);

console.log(num); // Output: 123

3. **Boolean()**: Converts a value to a boolean.

let num = 0;

let bool = Boolean(num);

console.log(bool); // Output: false

**In JavaScript, values are categorized as either "truthy" or "falsy"**

**Falsy Values:**

1. **false**: The boolean value false itself.
2. **0**: The number zero.
3. **""**: Empty string.
4. **null**: The absence of any value.
5. **undefined**: A variable that has not been assigned a value or a property that does not exist.
6. **NaN**: Not-a-Number.

**Truthy Values:**

1. **true**: The boolean value true itself.
2. **Non-zero numbers**: Any number other than 0 (including negative numbers and decimals).
3. **Non-empty strings**: Any string with at least one character.
4. **Non-empty arrays**: Arrays with at least one element.
5. **Objects**: Any object (including functions and arrays) is truthy, even if it's empty.
6. **Functions**: Any function is truthy, even if it doesn't return anything.

//program to find falsy and truthy values

var a = [];

var b = a ? true : false;

console.log(b);

