**Introduction to Javascript Operations**

**Assignment Java Script**

**Que 1 : Explain the role of operator in javascript . Why they are essential in programming ?**

**Ans :** Operators in JavaScript are symbols that perform operations on operands. Operands can be variables, values, or expressions. Operators are essential in programming because they allow us to manipulate data and perform calculations#

They are essential in programming languages as it can be used to

1. Perform Arithmetic operation.
2. Assign operators.
3. Compare Operators.
4. Perform logical operations.

**Que 2 : Describe the categorization of operator in javascript based based on their functionally .Provide example for each category?**

**Ans :** Operators in JavaScript are categorised based on their functionality into the following groups:

**1.Arithmetic Operators:** These operators perform mathematical operations on operands. Operands can be numbers or variables.

**Example :**

**// Addition (+)**

**const sum = 1 + 2; // sum is now equal to 3**

**// Subtraction (-)**

**const difference = 10 - 5; // difference is now equal to 5**

**// Multiplication (\*)**

**const product = 3 \* 4; // product is now equal to 12**

**// Division (/)**

**const quotient = 12 / 3; // quotient is now equal to 4**

**// Exponentiation (\*\*)**

**const power = 2 \*\* 3; // power is now equal to 8**

**// Modulo (%)**

**const remainder = 10 % 3; // remainder is now equal to 1**

**2 .Assignment Operators:** These operators assign values to variables

**// Assignment operator (=)**

**const myVariable = 10;**

**// Addition assignment operator (+=)**

**myVariable += 5; // myVariable is now equal to 15**

**// Subtraction assignment operator (-=)**

**myVariable -= 5; // myVariable is now equal to 10**

**// Multiplication assignment operator (\*=)**

**myVariable \*= 2; // myVariable is now equal to 20**

**// Division assignment operator (/=)**

**myVariable /= 2; // myVariable is now equal to 10**

**// Modulo assignment operator (%=)**

**myVariable %= 3; // myVariable is now equal to 1**

**3 . Comparison Operators:** These operators compare two values and return a Boolean value (true or false).

**Example : // Equal to (==)**

**const isEqual = 10 == 10; // isEqual is now equal to true**

**// Not equal to (!=)**

**const isNotEqual = 10 != 10; // isNotEqual is now equal to false**

**// Greater than (>)**

**const isGreaterThan = 10 > 5; // isGreaterThan is now equal to true**

**// Less than (<)**

**const isLessThan = 10 < 5; // isLessThan is now equal to false**

**// Greater than or equal to (>=)**

**const isGreaterThanOrEqual = 10 >= 10; // isGreaterThanOrEqual is now equal to**

**true**

**// Less than or equal to (<=)**

**const isLessThanOrEqual = 10 <= 5; // isLessThanOrEqual is now equal to false**

**4 .4. Logical Operators:** These operators perform logical operations on Boolean values .

**Example : // AND (&&)**

**const isAndTrue = true && true; // isAndTrue is now equal to true**

**const isAndFalse = true && false; // isAndFalse is now equal to false**

**// OR (||)**

**const isOrTrue = true || false; // isOrTrue is now equal to true**

**const isOrFalse = false || false; // isOrFalse is now equal to false**

**// NOT (!)**

**const isNotTrue = !true; // isNotTrue is now equal to false**

**const isNotFalse = !false; // isNotFalse is now equal to true**

**Q3. Differentiate between unary, and binary operators in JavaScript. Give examples of each.?**

**Ans :** Operators in JavaScript are categorized based on the number of operands they required into the following groups

n Unary operators Unary operators operate on a single operand

n Binary operators Binary operators operate on two operands.

Unary operators in JavaScript are:

1. Increment (++): Increases the value of a variable by 1.
2. Decrement (--): Decreases the value of a variable by 1.
3. Negation (-): Converts its operand to a number and negates it.
4. Logical NOT (!): Converts its operand to a boolean value and negates it.
5. TypeOf: Returns a string indicating the type of the operand.
6. Void: Evaluates an expression and returns undefined.
7. Bitwise NOT (~): Inverts the bits of its operand.

**Binary operators in JavaScript are :**

**+**

**\***

**/**

**%**

**==**

**===**

**!=**

**!==**

**>**

**<**

**>=**

**<=**

**&&**

**||**

**Q4. Discuss the precedence and associativity of operators in JavaScript. Why is understanding these**

**concepts important?**

**Ans :** Precedence refers to the order in which operators are evaluated when multiple operators are present in an expression. Operators with higher precedence are executed first. For example, in the expression 3 + 5 \* 2,

multiplication (\*) has higher precedence than addition (+), so the multiplication is performed first

Associativity refers to the order in which operators of the same precedence are evaluated. Some operators associate left to right, meaning they are evaluated from left to right. Others associate right to left, meaning they are evaluated from right to left.

Understanding precedence and associativity is important for several reasons

± Understanding the order of precedence ensures that expressions are

evaluated correctly, preventing potential errors or unexpected results

± When working with complex expressions, knowing the precedence and associativity

allows developers to write code that behaves as intended, making it easier to reason about and maintain

± In case of unexpected results, knowing operator precedence and

associativity helps in debugging and identifying the root cause of issues

± Knowledge of precedence and associativity can be used to optimize code by avoiding.