November 4th

Q 1. Explain the role of operators in JavaScript. Why are they essential in programming?

Operators in JavaScript are symbols that perform specific operations on values (operands). They are essential because they allow you to perform arithmetic, compare values, assign values, and carry out logic. Without operators, performing tasks like calculations, decisions, and data manipulation would be cumbersome.

Q 2. Describe the categorization of operators in JavaScript based on their functionality. Provide examples for each category.

1. **Arithmetic Operators**: Perform basic math operations.

- Example: `+`, `-`, `*`, `/`, `%`

  ```js
  let sum = 5 + 3;  // 8`
  ```

~~~~
2. **Assignment Operators**: Assign values to variables.

- Example: `=`, `+=`, `-=`, `*=`
    ``` js
    let x = 10; x += 5;  // x = 15`
    ```
3. **Comparison Operators**: Compare values and return `true` or `false`.

- Example: `==`, `===`, `!=`, `>`, `<`
    ```js
    let isEqual = (5 === 5);  // true`
~~~~

4. **Logical Operators**: Combine multiple conditions.

- Example: `&&`, `||`, `!`

```
let result = (5 > 3 && 2 < 4);  // true`
```

~~~
5. **Bit-wise Operators**: Work with binary numbers.

- Example: `&`, `|`, `^`, `~`
    ```js
    let result = 5 & 1;  // 1 (binary AND)`
~~~

6. **String Operators**: Concatenate strings.

- Example: `+`

```
let message = "Hello" + " World";  // "Hello World"`
```

Q 3. Differentiate between unary, binary, and ternary operators in JavaScript. Give examples of each.

1. **Unary Operators**: Work with a single operand.

- Example: `++`, `--`, `!`, `typeof`

```
 let x = 5; x++;  // x becomes 6 (increment)
```

2. **Binary Operators**: Work with two operands.

- Example: `+`, `-`, `*`, `==`, `&&`

```
let sum = 4 + 2;  // 6`
```

```
3. **Ternary Operator**: Uses three operands, often for conditional
operations.

- Example: `condition ? valueIfTrue : valueIfFalse`
```

~~~
```js
    let isEven = (x % 2 === 0) ? "Even" : "Odd";`
~~~

Q 4. Discuss the precedence and associativity of operators in JavaScript. Why is understanding these concepts important?

Precedence refers to the order in which operations are evaluated in an expression, and **associativity** defines the direction (left-to-right or right-to-left) in which operators of the same precedence level are evaluated.

- **Precedence Example**: Multiplication has higher precedence than addition.

```
   let result = 5 + 3 * 2;  // result is 11, not 16 (because * is done first)`
```

- **Associativity Example**: For operators like `=`, associativity is right-to-left.

  ```
  let a = b = c = 5;  // c is assigned 5, then b is assigned 5, then a is assigned 5`
  ```

~~~
### Q 5. Write a JavaScript program that calculates the simple interest using the formula Simple Interest = (principal * rate * time) / 100.



```js
let principal = 1000; let rate = 5; let time = 2;  let simpleInterest = (principal * rate * time) / 100; console.log("Simple Interest: " + simpleInterest);`
~~~

Explanation: We can calculate simple interest using the formula, where `principal`, `rate`, and `time` are input values.

Q 6. Write a JavaScript program to calculate the Body Mass Index (BMI) using the formula BMI = weight (kg) / (height * height).

```
let weight = 70; // in kg
let height = 1.75; // in meters
let bmi = weight / (height * height);
console.log("BMI: " + bmi);`
```

Q 7. Write a program in JavaScript to calculate the area of a circle given its radius value of 10. Use appropriate arithmetic operators.

```
let radius = 10; let area = Math.PI * radius * radius;
// or
Math.pow(radius, 2)
console.log("Area of the circle: " + area);`
```

Explanation: The area of a circle is calculated using the formula `π * radius²`. We use `Math.PI` for the value of π.