JS - Operators

1. Arithmetic Operators

Arithmetic operators are used to perform basic arithmetic operations on numbers.

```
// Arithmetic Operators in JavaScript:

// 1. Addition (+)
let sum = 5 + 3; // 8

// 2. Subtraction (-)
let difference = 5 - 3; // 2

// 3. Multiplication (*)
let product = 5 * 3; // 15

// 4. Division (/)
let quotient = 6 / 3; // 2

// 5. Modulus (%)
let remainder = 7 % 3; // 1

// 6. Exponentiation (**)
let power = 2 ** 3; // 8

/*
| Operator           | Example          | Description                                     |
|-----------------   |----------------- |------------------------------------------------ |
| Addition (+)       | 5 + 3 = 8        | Adds two values.                                |
| Subtraction (-)    | 5 - 3 = 2        | Subtracts the second value from the first.      |
| Multiplication (*) | 5 * 3 = 15       | Multiplies two values.                          |
| Division (/)       | 6 / 3 = 2        | Divides the first value by the second.          |
| Modulus (%)        | 7 % 3 = 1        | Returns the remainder of the division.          |
| Exponentiation (**)| 2 ** 3 = 8       | Raises the first value to the power of the second.|
*/
```

2. Assignment Operators

Assignment operators are used to assign values to variables.

```
// Assignment Operators in JavaScript:

// 1. Assignment (=)
let a = 5;  // Assigns the value 5 to a

// 2. Add and assign (+=)
a += 3;  // a = a + 3 -> 8

// 3. Subtract and assign (-=)
a -= 2;  // a = a - 2 -> 6

// 4. Multiply and assign (*=)
a *= 2;  // a = a * 2 -> 12

// 5. Divide and assign (/=)
a /= 4;  // a = a / 4 -> 3

// 6. Modulus and assign (%=)
a %= 2;  // a = a % 2 -> 1

/*
| Operator                 | Example         | Description                                        |
|-----------------         |-----------------|------------------------------------------------    |
| Assignment (=)           | a = 5           | Assigns the value to the variable.                 |
| Add and assign (+=)      | a += 3 -> 8     | Adds a value to the variable and assigns the result. |
| Subtract and assign (-=) | a -= 2 -> 6     | Subtracts a value and assigns the result.          |
| Multiply and assign (*=) | a *= 2 -> 12    | Multiplies a value and assigns the result.         |
| Divide and assign (/=)   | a /= 4 -> 3     | Divides a value and assigns the result.            |
| Modulus and assign (%=)  | a %= 2 -> 1     | Takes the modulus and assigns the result.          |
*/
```

3. Comparison Operators

Comparison operators are used to compare two values and return a boolean result (`true` or `false`).

```
// Comparison Operators in JavaScript:

// 1. Equal to (==)
let isEqual = 5 == '5'; // true (value is the same, type is different)

// 2. Strict equal to (===)
let isStrictEqual = 5 === '5'; // false (value and type must be the same)

// 3. Not equal to (!=)
let isNotEqual = 5 != 3; // true (values are different)

// 4. Strict not equal to (!==)
let isStrictNotEqual = 5 !== '5'; // true (value or type is different)

// 5. Greater than (>)
let isGreaterThan = 10 > 5; // true

// 6. Less than (<)
let isLessThan = 5 < 10; // true

// 7. Greater than or equal to (>=)
let isGreaterThanOrEqual = 5 >= 5; // true

// 8. Less than or equal to (<=)
let isLessThanOrEqual = 3 <= 5; // true

/*
| Operator                 | Example              | Description                                             |
|---------------------     |----------------------|-------------------------------------------------------  |
| Equal to (==)            | 5 == '5'             | Compares values for equality (ignores type).            |
| Strict equal to (===)    | 5 === '5'            | Compares both value and type for equality.              |
| Not equal to (!=)        | 5 != 3               | Compares values for inequality.                         |
| Strict not equal to (!==) | 5 !== '5'           | Compares both value and type for inequality.            |
| Greater than (>)         | 10 > 5               | Returns true if left value is greater than right.       |
| Less than (<)            | 5 < 10               | Returns true if left value is less than right.          |
| Greater or equal (>=)    | 5 >= 5               | Returns true if left value is greater or equal to right.|
| Less or equal (<=)       | 3 <= 5               | Returns true if left value is less or equal to right.   |
*/
```

4. Logical Operators

Logical operators are used to perform logical operations, typically on boolean values.

```
// Logical Operators in JavaScript:

// 1. AND (&&)
let isBothTrue = true && false;  // false

// 2. OR (||)
let isEitherTrue = true || false; // true

// 3. NOT (!)
let isNotTrue = !true; // false

/*
| Operator        | Example         | Description                                       |
|-----------------|-----------------|-------------------------------------------------- |
| AND (&&)        | true && false    | Returns true if both values are true.            |
| OR (||)         | true || false    | Returns true if at least one value is true.      |
| NOT (!)         | !true            | Reverses the boolean value (true becomes false). |
*/
```

5. Bitwise Operators

Bitwise operators work with the binary representation of numbers.

```
// Bitwise Operators in JavaScript:

// 1. AND (&)
let bitwiseAnd = 5 & 3;  // 1 (binary 101 & 011 = 001)

// 2. OR (|)
let bitwiseOr = 5 | 3;   // 7 (binary 101 | 011 = 111)

// 3. XOR (^)
let bitwiseXor = 5 ^ 3;  // 6 (binary 101 ^ 011 = 110)

// 4. NOT (~)
let bitwiseNot = ~5;     // -6 (binary ~101 = ...11111010)

// 5. Left Shift (<<)
let leftShift = 5 << 1;  // 10 (binary 101 << 1 = 1010)

// 6. Right Shift (>>)
let rightShift = 5 >> 1; // 2 (binary 101 >> 1 = 10)

/*
| Operator        | Example         | Description                                                 |
|-----------------|-----------------|------------------------------------------------------------ |
| AND (&)         | 5 & 3           | Performs a binary AND operation.                            |
| OR (|)          | 5 | 3           | Performs a binary OR operation.                             |
| XOR (^)         | 5 ^ 3           | Performs a binary XOR operation.                            |
| NOT (~)         | ~5              | Performs a binary NOT operation (inverts bits).             |
| Left Shift (<<) | 5 << 1          | Shifts bits to the left by a specified number of positions. |
| Right Shift (>>)| 5 >> 1          | Shifts bits to the right by a specified number of positions.|
*/
```

6. Unary Operators

Unary operators are operators that act on a single operand.

```
// Unary Operators in JavaScript:

// 1. Unary plus (+)
let unaryPlus = + '5'; // 5 (Converts string to number)

// 2. Unary negation (-)
let unaryNeg = - '5'; // -5 (Converts string to number and negates it)

// 3. Increment (++)
let increment = 5;
increment++;  // 6

// 4. Decrement (--)
let decrement = 5;
decrement--;  // 4

// 5. typeof
let type = typeof 5;  // "number"

// 6. delete
let obj = {name: "John"};
delete obj.name; // Removes the "name" property from the object

/*
| Operator           | Example             | Description                                         |
|-----------------   |---------------------|---------------------------------------------------- |
| Unary plus (+)     | + '5'               | Converts the operand to a number.                   |
| Unary negation (-) | - '5'               | Converts the operand to a number and negates it.    |
| Increment (++)     | i++                 | Increases the value by 1.                           |
| Decrement (--)     | i--                 | Decreases the value by 1.                           |
| typeof             | typeof 5            | Returns the type of the operand.                    |
| delete             | delete obj.name     | Deletes a property from an object.                  |
*/
```

7. Ternary Operator

The ternary operator is a shorthand for an `if-else` statement.

```
// Ternary Operator in JavaScript:

let result = (5 > 3) ? "Yes" : "No"; // "Yes"

/*
| Operator        | Example         | Description
|-----------------|-----------------|-------------------------------------------------------------------------
| Ternary (?)     | (5 > 3) ? "Yes" : "No" | Evaluates the condition; returns the first value if true, otherwis
*/
```