<!-- ! 2) Assignment Operator -->

Assignment operators in JavaScript are used to assign values to variables. The most basic assignment operator is the equal sign (`=`)

Types of Assignment Operators

1. Assignment (`=`)

- Assigns the value of the right operand to the left operand.
- Example:

```
let a = 5;    <!-- a is now 5 -->
```

2. Addition Assignment (`+=`)

- Adds the right operand to the left operand and assigns the result to the left operand.
- Example:

```
let a = 5;
a += 3;        <!-- a is now 8 -->
```

3. Subtraction Assignment (`-=`)

- Subtracts the right operand from the left operand and assigns the result to the left operand.
- Example:

```
let a = 5;
a -= 3;       <!--  a is now 2 -->
```

4. Multiplication Assignment (`*=`)

- Multiplies the left operand by the right operand and assigns the result to the left operand.
- Example:

```
let a = 5;
a *= 3;      <!-- a is now 15 -->
```

5. **Division Assignment (`/=`)**

- Divides the left operand by the right operand and assigns the result to the left operand.
- Example:

```
let a = 6;
a /= 3;       <!-- a is now 2 -->
```

6. **Remainder Assignment (`%=`)**

- Computes the remainder of dividing the left operand by the right operand and assigns the result to the left operand.
- Example:

```
let a = 5;
a %= 2;      <!-- a is now 1 -->
```

7. **Exponentiation Assignment (`=`)**

- Raises the left operand to the power of the right operand and assigns the result to the left operand.
- Example:

```
let a = 2;
a = 3;     <!-- a is now 8 -->
```

<!-- ! 3) Relational Operator -->

Relational operators in JavaScript are used to compare two values. They return a boolean value (`true` or `false`) based on whether the comparison is true or not. These operators are essential for making decisions in control structures like `if` statements, loops, and other conditional expressions.

Types of Relational Operators

1. **Greater than (`>`)**

   - Returns `true` if the left operand is greater than the right operand.
   - Example:

```
let a = 10;
let b = 5;
console.log(a > b);       <!-- true -->
```

2. **Greater than or equal to (`>=`)**

   - Returns `true` if the left operand is greater than or equal to the right operand.
   - Example:

```
let a = 10;
let b = 10;
console.log(a >= b); <!-- true -->
```

3. **Less than (`<`)**

   - Returns `true` if the left operand is less than the right operand.
   - Example:

```
let a = 5;
let b = 10;
console.log(a < b);   <!-- true -->
```

4. **Less than or equal to (`<=`)**

   - Returns `true` if the left operand is less than or equal to the right operand.
   - Example:

```
let a = 5;
let b = 5;
console.log(a <= b);   <!-- true -->
```

5. **Equal to (`==`)**

   - Returns `true` if the operands are equal. Performs type coercion if the operands are of different types.
   - Example:

```
let a = 5;
let b = '5';
console.log(a == b);   <!-- true  -->
```

6. **Strict equal to (`===`)**

   - Returns `true` if the operands are equal and of the same type. No type coercion is performed.
   - Example:

```
let a = 5;
let b = '5';
console.log(a === b);  <!-- false  -->
```

7. **Not equal to (`!=`)**

   - Returns `true` if the operands are not equal. Performs type coercion if the operands are of different types.
   - Example:

```
let a = 5;
let b = '5';
console.log(a != b);   <!-- false  -->
```

8. Strict not equal to (`!==`)

- Returns `true` if the operands are not equal or not of the same type. No type coercion is performed.
- Example:

```
let a = 5;
let b = '5';
console.log(a !== b);    <!-- true  -->
```

```
// !   Relational Operator

let num1 = 10 ;

let num2 = 20 ;


// !  1.  greater than (>)


console.log(num1 > num2)    // false

console.log(num2> num1)    // true

console.log('-------------------------------------------------')

// !  2. less than  (<)

console.log(num1 < num2 )   // true
console.log(num2 < num1)    // false

console.log('-------------------------------------------------')

// ! 3. greaterthan or equal to ( >=)

let num3 = 10
let num4 = 10

console.log(num3 >= num4)   // true
console.log('-------------------------------------------------')

// ! 4. Less than or equal to (<=)

console.log(num3 <= num4)     // true

console.log("-----------------------------------------")
```

```
// ! 5.   equal to (==)

let num5 = 20 ;
let num6 = '20'


console.log(num5 == num6) // true
console.log('-----------------------------------------------')

// ! 6. Strict  equal to (===)

console.log(num5 === num6)  // false

console.log('-----------------------------------------------')


// ! 7. not equal to  ( !=)

console.log( num5 != num6)     //false
console.log('-------------------------------------')

//! 8.   Strict not equal to ( !== )

console.log(num5 !== num6)   // true
```

<!-- ! 4) Logical Operator -->

Logical operators in JavaScript are used to perform logical operations and return a boolean result (`true` or `false`). These operators are typically used with boolean (logical) values, but they can also be applied to other types to return a boolean value.

Types of Logical Operators

1. Logical AND (`&&`)

- Returns `true` if both operands are `true`; otherwise, returns `false`.
- Example:

```
let a = true;
let b = false;
console.log(a && b);    <!-- false  -->
console.log(a && true); <!-- true  -->
```

2. Logical OR (`||`)

- Returns `true` if at least one of the operands is `true`; otherwise, returns `false`.
- Example:

```
let a = true;
let b = false;
console.log(a || b);     <!-- true  -->
console.log(b || false); <!-- false  -->
```

3. Logical NOT (`!`)

- Returns `true` if the operand is `false`; otherwise, returns `false`.
- Example:

```
let a = true;
let b = false;
console.log(!a); <!-- false  -->
console.log(!b); <!-- true  -->
```

```
// !  Logical Operator

let num1 = 10
let num2 = 20

// !  AND Operator

let ansForAnd1 =  (num1>10) && (num2<20)
console.log(ansForAnd1)                 //output : false && fasle => false

let ansForAnd2 = (num1==10)  &&  (num2>20)
console.log(ansForAnd2)                 // output : true && false => false

let ansForAnd3 = (num1==10)  &&  (num2>10)
console.log(ansForAnd3)                    // output : true && true => true

let ansForAnd4 = (num1>10)  &&  (num2>10)
console.log(ansForAnd4)                   // output : false && true => false

console.log("-----------------------------------------------------------------
--------------")


// !  OR Operator

let ansForOr1 = (num1>10) || (num2<20)
console.log(ansForOr1)                    //output : false || fasle => false

let ansForOr2 = (num1==10)  ||  (num2>20)
console.log(ansForOr2)                  // output : true || false => true

let ansForOr3 = (num1==10)  ||  (num2>10)
console.log(ansForOr3)                    // output : true || true => true

let ansForOr4 = (num1>10)  ||  (num2>10)
console.log(ansForOr4)                    // output : false || true => true
```