Operator precedence and associativity

Sure! Let's dive into **Operator Precedence** and **Associativity** in JavaScript — two key concepts that determine **how expressions are evaluated** when multiple operators are involved.

🧮 1. What is Operator Precedence?

Operator Precedence defines the **order** in which different operators in an expression are evaluated.

Higher precedence means the operator is executed **before** those with lower precedence.

📊 Example:

```
let result = 5 + 2 * 3;
```

Here:

- * (multiplication) has higher precedence than `+` (addition).
- So, `2 * 3` is evaluated **first**, then `5 + 6 = 11`.

🔁 2. What is Associativity?

Associativity defines the **direction** in which operators of the **same precedence** are evaluated — either **left to right** or **right to left**.

👉 Example: Left-to-Right Associativity

```
let result = 10 - 5 - 2;
```

All `-` operators have the same precedence and are **left-associative**, so:

- `10 - 5 = 5`
- Then `5 - 2 = 3`

👉 Example: Right-to-Left Associativity

```
let a = b = c = 10;
```

The assignment operator `=` is **right-associative**, so:

- `c = 10`
- `b = 10`
- `a = 10`

📋 Operator Precedence Table (Simplified)

Precedence	Operator(s)	Type	Associativity
1 (Highest)	`()`	Grouping	N/A
2	`++`, `--`	Unary	Right-to-left
3	`*`, `/`, `%`	Arithmetic	Left-to-right
4	`+`, `-`	Arithmetic	Left-to-right
5	`<`, `<=`, `>`, `>=`	Comparison	Left-to-right
6	`==`, `!=`, `===`, `!==`	Equality	Left-to-right
7	`&&`	Logical AND	Left-to-right
8	`		`
9	`?:`	Ternary	Right-to-left
10	`=`, `+=`, `-=` etc.	Assignment	Right-to-left
11 (Lowest)	`,`	Comma	Left-to-right

✅ Practical Example

```
let x = 10 + 5 * 2 ** 2;
```

Step-by-step:

1. `*` has highest among arithmetic → `2 ** 2 = 4`
2. Then → `5 * 4 = 20`
3. Then `+` → `10 + 20 = 30`

✅ Result: `x = 30`

💡 Tips to Avoid Confusion

- **Use parentheses** `()` to control the evaluation order and improve readability.
- Learn precedence only for **commonly used operators**; look up the rest when needed.

📌 Summary

Concept	Explanation
Precedence	Which operator runs first
Associativity	Direction of evaluation for same-precedence operators
Best Practice	Use parentheses `()` for clarity and control

Would you like a **mini quiz** or a **cheat sheet** for quick reference?