***precedence and associativity of operators in JavaScript.***

**🡪Precedence :** Precedence determines the order of evaluation when multiple operators are present in an expression. Operators with higher precedence are evaluated first.

* Example:

Let prec = 3 + 5 \* 2; // The result will be 13, not 16

**🡪 Associativity:** Associativity determines the order in which operators of the same precedence are evaluated. It can be left-to-right or right-to-left.

* Example:

Let asso = 10 + 5 + 2; // The result will be 17

🡪Evulate on the bases of following:

* Grouping: () - Highest precedence, left-to-right associativity.
* Member Access: . and [] - Left-to-right associativity.
* Computed Member Access: [ ] - Left-to-right associativity.
* Function Call: () - Left-to-right associativity.
* New (with arguments): new - Right-to-left associativity.
* Unary Operators: +, -, !, typeof, void, delete, ++, -- (prefix) - Right-to-left associativity.
* Exponentiation: \*\* - Right-to-left associativity.
* Multiplication/Division/Modulus: \*, /, % - Left-to-right associativity.
* Addition/Subtraction: +, - - Left-to-right associativity.
* Bitwise Shift: <<, >>, >>> - Left-to-right associativity.
* Relational: <, <=, >, >=, instanceof, in - Left-to-right associativity.
* Equality: ==, ===, !=, !== - Left-to-right associativity.
* Bitwise AND: & - Left-to-right associativity.
* Bitwise XOR: ^ - Left-to-right associativity.
* Bitwise OR: | - Left-to-right associativity.
* Logical AND: && - Left-to-right associativity.
* Logical OR: || - Left-to-right associativity.
* Conditional (Ternary) Operator: ? : - Right-to-left associativity.