Operator Precedence in JavaScript

Introduction to Operator Precedence

In this lecture, we discuss the precedence of different operators in JavaScript. Operator precedence determines the order in which operations are executed in an expression.

Revisiting Previous Code

We begin by bringing back some code from the last lecture, specifically the calculation of ages and the code discussed at the end of the previous video.

The Precedence Question

The question posed is: Why are the two subtractions executed before the comparison operator in the following expression?

javascript Code Sample

ageJonas - 1991 > ageSarah - 2018

It works this way because JavaScript has a well-defined order of operator precedence, which is the order in which operators are executed.

Exploring the Precedence Table

To see the precedence of all the different operators, we refer to the MDN (Mozilla Developer Network) operator precedence table. MDN is a widely used documentation site, and we will use it throughout the course.

On the table, grouping with parentheses has the highest precedence of 21. Operators like plus and minus have a precedence of 17. There are other operators such as **typeof**, exponentiation, and various math and comparison operators.

This table is a useful reference to see all the operators that exist in JavaScript in one place.

Operator Precedence in Practice

Let us focus on our example and understand what happens in terms of operator precedence. The two calculations (subtractions) are done before the comparison. In the table, the comparison operator (**>**) has a lower precedence than subtraction.

Precedence Numbers

Subtraction has a precedence of 14, while the comparison operator has 12. You do not need to memorize these numbers; it is more important to have a general idea of which operators are executed first. Usually, math operators are executed before comparison operators.

Operator Associativity: Left-to-Right and Right-to-Left

The table also shows which operators are executed from left to right and which from right to left. For example, the exponentiation operator is executed from right to left, while most mathematical operators are executed from left to right.

Example: Left-to-Right Execution

Consider the following calculation:

javascript Code Sample

25 - 10 - 5

This should result in 10. If the operation were right to left, the result would be different. This demonstrates left-to-right execution for subtraction.

Example: Right-to-Left Execution (Assignment)

Assignment is an example of right-to-left execution. Assignment has one of the lowest precedences.

javascript Code Sample

let x, y;

We can declare two variables at the same time. Both x and y are now undefined.

javascript Code Sample

x = y = 25 - 10 - 5;

javascript Code Sample

console.log(x, y);

Both x and y will be 10. Let us analyze why this happens. JavaScript first executes the minus operators, which have higher precedence than assignment. After the subtraction, we have:

javascript Code Sample

x = y = 10;

Now, only the assignment operators are left, which are executed right to left. First, y is assigned 10, then x is assigned the value of y, which is now 10. Thus, both x and y become 10.

If assignment were left to right, x would be assigned the value of y (which is undefined), and then y would be assigned 10. This is not the result we expect.

The Importance of Grouping (Parentheses)

Grouping using parentheses has the highest precedence. Operations within parentheses are executed first, just like in mathematics.

Example: Calculating the Average

Suppose we have two variables, ageJonas and ageSarah. To calculate the average age, we add the two ages and divide by two.

javascript Code Sample

console.log(ageJonas, ageSarah);

javascript Code Sample

averageAge = ageJonas + ageSarah / 2;

According to the precedence table, division happens before addition. Thus, ageSarah is divided by two first, then ageJonas is added. This can lead to an incorrect result.

javascript Code Sample

console.log(averageAge);

If the result does not make sense, use parentheses to group the addition first:

javascript Code Sample

averageAge = (ageJonas + ageSarah) / 2;

Now, the addition is executed first, then the division, giving the correct average.

Conclusion

Operator precedence and associativity are crucial for understanding how JavaScript evaluates expressions. Use parentheses to ensure the intended order of operations.

Key Takeaways

- JavaScript executes operators according to a well-defined precedence order, which determines the sequence in which operations are performed.
- Mathematical operators generally have higher precedence than comparison or assignment operators.
- Operator associativity (left-to-right or right-to-left) affects how expressions are evaluated, especially for operators like assignment and exponentiation.
- Parentheses can be used to group operations and override the default precedence, ensuring correct calculation order.