

Session 1

Preliminaries

Javascript is a high-level, interpreted programming language that is widely used for web development. It was created by Brendan Eich in 1995 and has since become one of the most popular programming languages in the world. JavaScript is primarily used for client-side scripting, allowing developers to create interactive and dynamic web pages. It can also be used on the server-side with environments like Node.js, Deno, Bun and Cloudflare Workers (don't mix with web workers).

In this session, we will introduce the basic JavaScript constructs, conditional statements and control flow statements. The lab manual includes good examples and exercises to practice these concepts.



Suggestion

Before trying the solutions provided for the exercises, try to solve them on your own. This will help you understand the concepts better and improve your problem-solving skills.

Also, Lab manual includes great examples before exercises, make sure to go through them as well.

Objectives

✗ Do not copy. Read for understanding and the viva

The objectives of this session are to:

- Understand the basic JavaScript constructs
- Understand conditional statements and control flow statements

Arithmetic Operators

Problem Statement

■ Write in lab record

Q1: Write a JavaScript code to perform the two digit arithmetic operations: Addition, Subtraction, Multiplication and Division.

Approach

✖ Do not copy. Read for understanding and the viva

To perform arithmetic operations on two digits, we can follow these steps:

- Define two variables to hold the digits we want to operate on.
- Use the arithmetic operators (`+` , `-` , `*` , `/`) to perform the desired operations.
- Print the results to the console.

Code Implementation

■ Write in lab record

arithmetic-operation.js

```
1 // 1. Define the input numbers
2 // We use 'let' or 'const' to store our numeric values
3 const num1 = 15;
4 const num2 = 10;
5
6 console.log(`Operating on numbers: ${num1} and ${num2}\n`);
7
8 // 2. Addition (+)
9 // The sum of two numbers
10 const addition = num1 + num2;
11 console.log("Addition: " + num1 + " + " + num2 + " = " + addition);
12
13 // 3. Subtraction (-)
14 // Subtracting the second number from the first
15 const subtraction = num1 - num2;
16 console.log("Subtraction: " + num1 + " - " + num2 + " = " + subtraction);
17
18 // 4. Multiplication (*)
19 // The product of two numbers
20 const multiplication = num1 * num2;
21 console.log("Multiplication: " + num1 + " * " + num2 + " = " + multiplication);
22
23 // 5. Division (/)
24 // Dividing the first number by the second
25 // Note: We check if num2 is zero when taking input to avoid mathematical error
26 const division = num1 / num2;
27 console.log("Division: " + num1 + " / " + num2 + " = " + division);
```

Explanation

✗ Do not copy. Read for understanding and the viva

- **Variables:** We use `const` (constant) because these values aren't changing during the script's execution. If you want to change them later, you would use `let`.
- **Arithmetic Operators:** JavaScript uses standard mathematical symbols: `+`, `-`, `*`, and `/`.
- **Console.log:** This is the standard method to print output to the debugging console.
- **Division Safety:** In computer science, "Division by Zero" is a common logical error. Including a conditional `if` check demonstrates good programming practice for your lab submission.

- **String Concatenation:** Using the `+` symbol between a string (text in quotes) and a variable allows us to combine them into a single readable sentence for the output.

Pattern Printing

Problem Statement

■ Write in lab record

Q2: Write a JavaScript code to print 1 for 1 time 2 for 2 times, 3 for 3 times and so on up to 10.

Approach

✗ Do not copy. Read for understanding and the viva

The goal is to create a visual pattern where each number `n` repeats itself `n` times on a single line. To achieve this, we need Nested Loops:

- **The Outer Loop** will act as a counter that moves from 1 to 10. This tells us which number we are currently processing.
- **The Inner Loop** will run based on the value of the outer loop. If the outer loop is at 5, the inner loop will execute 5 times to print the digit "5".
- We will collect the digits in a string for each row so they stay on the same line, then print that row before moving to the next number.

Code Implementation

■ Write in lab record

pattern-printing.js

```
1 // Outer loop: Determines the number to be printed (1 through 10)
2 for (let i = 1; i ≤ 10; i++) {
3   let row = ""; // Reset the row string for the current number
4
5   // Inner loop: Controls the frequency of the print
6   // It runs 'i' times for the number 'i'
7   for (let j = 1; j ≤ i; j++) {
8     row += i + " "; // Append the number and a space to the row
9   }
10
11  // Print the completed row to the console
12  console.log(row.trim());
13 }
```

Explanation

✖ Do not copy. Read for understanding and the viva

- **Nested Loop Logic:** Think of this like a clock. The outer loop is the “hour” hand (moving slowly), and the inner loop is the “minute” hand (completing a full cycle for every single step the hour hand takes).
- **Dynamic Range:** The inner loop condition `j ≤ i` is what makes the pattern triangular. Because the limit of the inner loop depends on the current value of the outer loop, the rows get progressively longer.
- **String Accumulation:** In JavaScript, `console.log()` automatically adds a new line. To get numbers like `2 2` on the same line, we must first build a string (`rowOutput`) and print it once the inner loop finishes its work for that row.
- **trim() Method:** We use `.trim()` at the end to remove the very last space of each row, ensuring the output is clean for your lab record.

Portfolio Exercise

Q3: Discuss first with the group

 Edit this page

< Previous
Section 2 · Web Design

Next >
Session 2