Conditional and Loops - JavaScript

1. What are conditional statements? Explain conditional statements with syntax and examples.

   Conditional statements are programming constructs that allow a program to make decisions and execute different blocks of code based on whether a given condition is true or false. Essentially, they control the flow of execution in a program.

   Syntax and Structure:

   1. If statement: An if-statement executes a block of code only if the condition is true.

      EXAMPLE:

      ```
      let x = 10;
      if (x > 5) {
          console.log("x is greater than 5");
      }
      ```

      Syntax: if(condition):

   2. If-else statement: An if-else statement provides an alternative block of code to execute if the condition is false.

      EXAMPLE:

      ```
      let age = 17;
      if (age >= 18) {
          console.log("You are an adult.");
      } else {
          console.log("You are a minor.");
      }
      ```

      Syntax: if (condition):
      Else:

3. The if-elif-else statement: The if-elif-else statement allows for checking multiple conditions sequentially. The first condition that evaluates to true will have its corresponding code block executed, and the entire statement will then exit.

   EXAMPLE:

```
let score = 85;
if (score >= 90) {
    console.log("Grade: A");
} else if (score >= 80) {
    console.log("Grade: B"); // Output: Grade: B
} else if (score >= 70) {
    console.log("Grade: C");
} else {
    console.log("Grade: F");
}
```

   Syntax: if (condition):
   Elif (condition):
   else:

---

2. Write a program that grades students based on their marks
   - If greater than 90 then A Grade
   - If between 70 and 90 then a B grade
   - If between 50 and 70 then a C grade
   - Below 50 then an F grade

Program:

```
function gradecalculator(marks) {
    if (marks < 0 || marks > 100) {
        return "Invalid marks: Marks should be between 0 and 100.";
    }

    if(marks > 90) {
        return "A Grade";
    } else if(marks >= 70){
        return "B Grade";
    } else if(marks >= 50){
        return "C Grade";
    } else {
```

```
        return "F Grade";
    }
}
console.log("Marks 95:" + gradecalculator(95));
console.log("Marks 80:" + gradecalculator(80));
console.log("Marks 60:" + gradecalculator(60));
```

---

3. What are loops, and what do we need? Explain different types of loops with their syntax and examples.

   Loops are fundamental programming constructs that allow you to execute a block of code repeatedly based on a certain condition or for a specific number of times. We need loops to automate repetitive tasks, iterate over collections of data, and build efficient algorithms. Without loops, you'd have to write the same code multiple times, which is inefficient and prone to errors.

   For Loop:

   It's ideal for iterating over a sequence of numbers or a range of numbers.

   SYNTAX:

   for (initialization; condition; increment/decrement) {

   }

   EXAMPLE:

   for (let i = 0; i < 5; i++) {

   console.log(i);

   }

   While Loop:

   A while loop is used when you don't know the number of iterations beforehand, but the loop needs to continue as long as a certain condition remains TRUE. The condition is checked before each iteration.

   SYNTAX:

   While (condition){ }

   EXAMPLE:

```
let i = 0;
while (i < 5) {
    console.log(i);
    i++;
}
```

Do-While Loop:

A do-while loop is similar to a while loop, but with one key difference: the loop body is executed at least once before the condition is evaluated. If the condition is True, the loop continues.

SYNTAX:

```
Do{
}while (condition);
```

EXAMPLE:

```
let i = 0;
do {
    console.log(i);
    i++;
} while (i < 5);
```

---

4. Generate numbers between any 2 given numbers.
    - const num1 = 10:
    - const num2 = 25;
    - **Output**: 11, 12, 13, …., 25

PROGRAM:

```
function NumberBetween (num1,num2) {
    const numbers = [];
        if (typeof num1 !== 'number' || typeof num2 !== 'number' ||
isNaN(num1) || isNaN(num2)) {
        console.error("Error: Both inputs must be valid numbers.");
```

```
        return [];
    }
        if (num1 >= num2) {
        console.error("Error: The first number must be less than the
second number.");
        return [];
    }
        for (let i = num1 + 1; i <= num2; i++) {
        numbers.push(i);
    }
    return numbers;
}
const num1 = 10;
const num2 = 25;

const result = NumberBetween(num1, num2);
console.log(`Numbers between ${num1} and ${num2}:`);
if (Array.isArray(result)) {
    console.log(result.join(', '));
} else {
    console.log(result);
}
```

---

5. Use the while loop to print numbers from 1 to 25 in ascending and descending order.

   PROGRAM:

```
console.log("Numbers in Ascending Order (1 to 25):")
let i = 1;
while(i<=25){
    console.log(i);
    i++;
}

console.log("Numbers in Descending order (25 to 1):")
let j = 25;
```

```
while(j>=1){
    console.log(j);
    j--;
}
```
