Solution 1: In JavaScript, conditional statements are used to perform different actions based on different conditions. The most common types of conditional statements are if, else if, else, and switch.

1. if Statement

The if statement is used to execute a block of code if a specified condition is true.

Syntax:

```
if (condition) {
  // Code to be executed if the condition is true
}
```

Example:

```
let age = 20;

if (age >= 18) {
  console.log("You are eligible to vote.");
}
```

2. else Statement

The else statement is used to execute a block of code if the condition in the if statement is false.

Syntax:

```
if (condition) {
  // Code to be executed if the condition is true
} else {
  // Code to be executed if the condition is false
}
```

Example:

```
let age = 16;

if (age >= 18) {
  console.log("You are eligible to vote.");
} else {
  console.log("You are not eligible to vote.");
}
```

3. else if Statement

The else if statement is used to specify a new condition to test if the first condition is false.

Syntax:

```
if (condition1) {
  // Code to be executed if condition1 is true
} else if (condition2) {
  // Code to be executed if condition2 is true
} else {
  // Code to be executed if both conditions are false
}
```

Example:

```
let score = 85;

if (score >= 90) {
  console.log("Grade: A");
} else if (score >= 80) {
  console.log("Grade: B");
} else {
  console.log("Grade: C");
}
```

4. switch Statement

The switch statement is used to perform different actions based on different conditions (cases).

Syntax:

```
switch (expression) {
  case value1:
    // Code to be executed if expression === value1
    break;
  case value2:
    // Code to be executed if expression === value2
    break;
  // Add more cases as needed
  default:
    // Code to be executed if expression doesn't match any case
}
```

Example:

```
let day = 3;
let dayName;

switch (day) {
  case 1:
    dayName = "Sunday";
    break;
  case 2:
    dayName = "Monday";
    break;
  case 3:
    dayName = "Tuesday";
    break;
  case 4:
    dayName = "Wednesday";
    break;
  case 5:
    dayName = "Thursday";
    break;
  case 6:
    dayName = "Friday";
    break;
  case 7:
    dayName = "Saturday";
    break;
  default:
    dayName = "Invalid day";
}

console.log(dayName); // Output: Tuesday
```

Solution 2:

```
   function gradeStudent(marks) {

    let grade;

     if (marks > 90) {

        grade = 'A';

    } else if (marks >= 70 && marks <= 90) {

        grade = 'B';

    } else if (marks >= 50 && marks < 70) {

        grade = 'C';

    } else {

        grade = 'F';

    }

    return `Your grade is ${grade}.`;

}



// Example usage:

let studentMarks = 85;

console.log(gradeStudent(studentMarks));  // Output: Your grade is B.
```

Solution 3: Loops are a fundamental programming concept used to repeatedly execute a block of code as long as a specified condition is true. They help automate repetitive tasks, manage collections of data, and control the flow of a program efficiently. In JavaScript, there are several types of loops:

1. For Loop

The for loop is used when the number of iterations is known beforehand.

Syntax:

```
for (initialization; condition; increment/decrement) {
   // Code to be executed
}
```

Example:

```
for (let i = 0; i < 5; i++) {
   console.log(i);
}
// Output: 0 1 2 3 4
```

2. While Loop

The while loop is used when the number of iterations is not known and depends on a condition.

Syntax:

```
while (condition) {
   // Code to be executed
}
```

Example:

```
let i = 0;
while (i < 5) {
   console.log(i);
   i++;
}
// Output: 0 1 2 3 4
```

3. Do...While Loop

The do...while loop is similar to the while loop, but it guarantees that the code block is executed at least once, because the condition is evaluated after the code block.

Syntax:

```
do {
   // Code to be executed
} while (condition);
```

Example:

```
let i = 0;
do {
    console.log(i);
    i++;
} while (i < 5);
// Output: 0 1 2 3 4
```

4. For...In Loop

The for...in loop is used to iterate over the enumerable properties of an object.

Syntax:

```
for (let key in object) {
    // Code to be executed
}
```

Example:

```
let person = { name: 'John', age: 30, city: 'New York' };
for (let key in person) {
    console.log(key + ': ' + person[key]);
}
// Output: name: John  age: 30  city: New York
```

5. For...Of Loop

The for...of loop is used to iterate over the values of iterable objects like arrays, strings, or other collections.

Syntax:

```
for (let value of iterable) {
    // Code to be executed
}
```

Example:

```
let numbers = [1, 2, 3, 4, 5];
for (let number of numbers) {
    console.log(number);
}
// Output: 1 2 3 4 5
```

Solution 4:

```
    function generateNumbers(start, end) {

    let numbers = []

    if (start <= end) {

        for (let i = start; i <= end; i++) {
            numbers.push(i);

        }

    } else {

        console.log('Start number must be less than or equal to end number.');

    }

     return numbers;

}
```

Solution 5:

Ascending Order

To print numbers from 1 to 25 in ascending order:

```
let i = 1;

while (i <= 25) {
    console.log(i);
    i++;
}
```

Descending Order

To print numbers from 25 to 1 in descending order:

```
let i = 25;

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