The switch statement is a control structure that allows you to execute different blocks of code based on the value of a variable or expression.

Syntax

switch (expression) {

case value1:

// Code to execute if expression === value1

break;

case value2:

// Code to execute if expression === value2

break;

// Additional cases...

default:

// Code to execute if no cases match

}

example

const dayNumber = 6; // Change this number to test different days

switch (dayNumber) {

case 1:

console.log("Monday");

break;

case 2:

console.log("Tuesday");

break;

case 3:

console.log("Wednesday");

break;

case 4:

console.log("Thursday");

break;

case 5:

console.log("Friday");

break;

case 6:

console.log("Saturday");

break;

case 7:

console.log("Sunday");

break;

default:

console.log("Invalid day number. Please enter a number between 1 and 7.");

}

**Nested If Statements in JavaScript**

A nested if statement is an if statement placed inside another if statement. This allows you to check multiple conditions in a structured way. It can be useful when you need to evaluate a condition that depends on the outcome of a previous condition.

if (condition1) {

// Code to execute if condition1 is true

if (condition2) {

// Code to execute if condition2 is true

} else {

// Code to execute if condition2 is false

}

} else {

// Code to execute if condition1 is false

}

Example

let age = 20;

let hasID = true;

if (age >= 18) {

console.log("You are an adult.");

if (hasID) {

console.log("You can enter the club.");

} else {

console.log("You need an ID to enter the club.");

}

} else {

console.log("You are not an adult.");

}

### Ternary Operator in JavaScript

The ternary operator is a shorthand way to write an if...else statement. It is often used for simple conditional expressions and allows you to assign a value based on a condition in a more concise manner.

Syntax

condition ? expressionIfTrue : expressionIfFalse;

* **condition**: The expression to evaluate (it should return a boolean value).
* **expressionIfTrue**: The value to return if the condition is true.
* **expressionIfFalse**: The value to return if the condition is false.

**Example 1**

let age = 18;

let canVote = (age >= 18) ? "You can vote." : "You cannot vote.";

console.log(canVote);

**Example 2**

let score = 85;

let grade = score >= 90 ? "A" : score >= 80 ? "B" : score >= 70 ? "C" : "D";

console.log(grade); // Output: "B"

loop statement

a **loop statement** is used to repeatedly execute a block of code as long as a specified condition is met.

Loops are essential for automating repetitive tasks, iterating over arrays, or processing data. JavaScript provides several types of loop statements:

### 1. for****Loop****

The for loop is commonly used when you know how many times you want to iterate.

for (initialization; condition; increment/decrement) {

// Code to execute

}

Example

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

console.log(i); // Output: 0, 1, 2, 3, 4

}

#### Explanation:

* **Initialization**: let i = 0 (executed once at the start).
* **Condition**: i < 5 (checked before each iteration; if true, the loop continues).
* **Increment/Decrement**: i++ (executed after each iteration)

### 2. while****Loop****

The while loop executes a block of code as long as a specified condition is true.

while (condition) {

// Code to execute

}

let i = 0;

while (i < 5) {

console.log(i); // Output: 0, 1, 2, 3, 4

i++;

}

### 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, even if the condition is false

do {

// Code to execute

} while (condition);

Example

let i = 0;

do {

console.log(i); // Output: 0, 1, 2, 3, 4

i++;

} while (i < 5);