JavaScript Conditional Statements and Looping Statements — Complete Guide for Beginners

Welcome! 🎯If you're new to JavaScript, mastering **conditional statements** and **loops** is a crucial step. These are the building blocks of logic and flow control in programming.

🧠1. Conditional Statements

💚Definition

Conditional statements allow your program to make **decisions**. They help your code run **different blocks of code based on conditions**.

In simple terms:

> "If something is true, do this. Otherwise, do that."

😾Syntax

```
if (condition) {
  // Code runs if condition is true
} else if (anotherCondition) {
  // Code runs if anotherCondition is true
} else {
  // Code runs if none of the above are true
}
```

You can also use the **ternary operator** (short form of if-else):

```
condition ? expressionIfTrue : expressionIfFalse;
```

❤️When to Use Conditional Statements

Use them when you want to **control the flow** of your program based on **a condition** (comparison, user input, status check, etc.).

👎🏿Examples

Example 1: Basic If Condition

```
let age = 20;
if (age >= 18) {
  console.log("You are eligible to vote.");
}
```

Real-life analogy: Checking age before voting.

Example 2: If-Else

```
let temperature = 25;
if (temperature > 30) {
  console.log("It's hot outside!");
} else {
  console.log("Weather is nice.");
}
```

Used in weather apps to show messages based on temperature.

Example 3: If-Else If-Else Chain

```
let marks = 85;
if (marks >= 90) {
  console.log("Grade A");
} else if (marks >= 75) {
  console.log("Grade B");
} else {
  console.log("Grade C");
}
```

Used in student grading systems.

Example 4: Ternary Operator

```
let isLoggedIn = true;
console.log(isLoggedIn ? "Welcome back!" : "Please log in.");
```

Used in login systems for user experience.

Example 5: Nested If

```
let userType = "admin";
let isActive = true;
if (userType === "admin") {
  if (isActive) {
    console.log("Admin is active");
  } else {
    console.log("Admin is inactive");
  }
}
```

Used in user management systems for admin validation.

🎉Project-Level Example (Conditional)

```
// Online shopping discount system
let cartValue = 500;
let hasCoupon = true;

if (cartValue > 1000) {
  console.log("You get a 20% discount!");
} else if (hasCoupon) {
  console.log("You get a 10% discount with your coupon!");
} else {
  console.log("No discount available.");
}
```

Used in e-commerce checkout logic.

Interview-Level Practice Questions

1. Difference between `==` and `===` in JavaScript.
2. What will be the output of this code?

```
let x = 0;
if (x) console.log('True'); else console.log('False');
```

3. Can we use `if` without `else`?
4. What is the ternary operator and where should it be used?
5. Explain short-circuit evaluation in conditions.

😌2. Looping Statements

💚Definition

Loops are used to **repeat a block of code multiple times** until a condition is false.

In short:

> “Do something again and again until I tell you to stop.”

😺Types and Syntax

1. `for` **loop**

```
for (initialization; condition; increment) {
  // code block to execute
}
```

2. `while` **loop**

```
while (condition) {
  // code block
}
```

3. `do...while` **loop**

```
do {
  // code block
} while (condition);
```

4. `for...of` **loop**

```
for (let item of iterable) {
  // code block
}
```

5. `for...in` **loop**

```
for (let key in object) {
  // code block
}
```

❤️When to Use Loops

Loop Type	Use Case
for	When the number of iterations is known
while	When the number of iterations is unknown but based on a condition
do...while	When code must run at least once before checking condition
for...of	When iterating through arrays or strings
for...in	When iterating through object properties

👎Examples

Example 1: For Loop

```
for (let i = 1; i <= 5; i++) {
  console.log("Count: " + i);
}
```

Counts numbers — used in pagination or iteration tasks.

Example 2: While Loop

```
let i = 1;
while (i <= 5) {
  console.log("Number: " + i);
  i++;
}
```

Used when you don't know how many times a loop will run, like waiting for user input.

Example 3: Do-While Loop

```
let x = 0;
do {
  console.log("Value: " + x);
  x++;
} while (x < 3);
```

Ensures code runs at least once — used in retry systems.

Example 4: For...of Loop

```
let fruits = ["Apple", "Banana", "Cherry"];
for (let fruit of fruits) {
  console.log(fruit);
}
```

Used to iterate arrays and strings easily.

Example 5: For...in Loop

```
let person = { name: "Alice", age: 25, city: "New York" };
for (let key in person) {
  console.log(key + ": " + person[key]);
}
```

Used for looping through object properties.

🎉Project-Level Example (Looping)

```
// Displaying products in an e-commerce site
const products = ["Laptop", "Mobile", "Watch", "Tablet"];

for (let i = 0; i < products.length; i++) {
  console.log(`Product ${i + 1}: ${products[i]}`);
}
```

Used in rendering product lists dynamically.

Interview-Level Practice Questions

1. What's the difference between `for`, `while`, and `do...while` loops?
2. How to break or continue a loop?
3. What is the difference between `for...of` and `for...in` loops?
4. How can you loop over an array of objects?
5. How do infinite loops occur and how can you prevent them?

🐱Real-Time Example: Combining Conditionals and Loops

```
// Checking for even and odd numbers in a range
for (let i = 1; i <= 10; i++) {
```

```
  if (i % 2 === 0) {
    console.log(i + " is even");
  } else {
    console.log(i + " is odd");
  }
}
```

Used in data validation, game logic, and backend filtering systems.

😢Mini Project Example

Title: ATM Withdrawal System

```
let balance = 5000;
let withdrawAmount = 2000;

if (withdrawAmount <= balance) {
  for (let i = 3; i > 0; i--) {
    console.log(`Processing... ${i}`);
  }
  balance -= withdrawAmount;
  console.log(`Withdrawal successful! New balance: ${balance}`);
} else {
  console.log("Insufficient funds!");
}
```

Combines conditions and loops — real-world banking simulation.

😖Key Takeaways

- **Conditionals** = Decision making
- **Loops** = Repeating actions
- Always ensure your conditions eventually become false in loops.
- Test your code with edge cases (like 0, empty arrays, null values).

😺Practice Tasks

1. Write a program that prints all odd numbers from 1 to 50.
2. Use an `if-else` statement to check if a number is positive, negative, or zero.
3. Loop through an array of student names and greet each one.
4. Create a grading system using conditions.
5. Combine loops and conditions to print only numbers divisible by both 3 and 5.

Final Note

Mastering **conditionals** and **loops** will make you confident in logic building. Once you understand these well, moving to **functions**, **arrays**, and **objects** becomes easier.

> Practice daily — write small programs until logic becomes second nature!