JavaScript `if` Statement: A Guide

Introduction

In JavaScript, the `if` statement is a vital control structure that allows you to execute specific blocks of code based on whether a given condition is true or false. It serves as the foundation for creating decision-making logic in your scripts.

what is if:

The `if` statement in JavaScript allows you to execute a block of code based on the evaluation of an expression.

- a conditional statement that allows you to execute code based on the value of an expression.
- it checks whether or not a condition evaluates as true, and executes a block of statements accordingly.

Basic `if` Statement

The basic `if` statement is a fundamental programming concept that allows the computer to make decisions based on the truth or falsity of a condition. It consists of an `if`keyword, a condition, and a code block. The condition is typically a comparison between two values or a Boolean expression that evaluates to either `True` or `False`.

Example 1: Checking Temperature

```
let temperature = 25;

if (temperature > 30) {
  console.log("It's a hot day!");
}
```

- Checks if the temperature is greater than 30.
- If true, prints "It's a hot day!";

Example 2: Checking If a Number is Positive

```
let number = -5;

if (number > 0) {
  console.log("The number is positive.");
}
```

- Verifies if the number is positive.
- Prints "The number is positive." if true;

Example 3: Checking if a String is Empty

```
let message = "";

if (message !== "") {
  console.log("The message is not empty.");
}
```

- Examines if the string `message` is empty.
- Prints "The message is not empty."

`if-else` Statement

The `"if"` part of the statement checks a condition, such as the value of a variable or the result of a function.
If the condition is true, the program executes the code block associated with the
If the condition is false, it will execute any code block that follows immediately after the `"else"` keyword.

Example 1: Greeting Based on Hour

```
let hour = 15;

if (hour < 12) {
  console.log("Good morning!");
} else {
  console.log("Good afternoon or evening!");
}
```

- `hour` is evaluated to be 15.
- The `if` condition checks if `hour` is less than 12 (morning). Since 15 is not less than 12, it moves to the `else` block.
- The `else` block is executed, printing "Good afternoon or evening!"
  Example 2: Checking if a Number is Even or Odd

```
let num = 7;

if (num % 2 === 0) {
  console.log("The number is even.");
} else {
  console.log("The number is odd.");
}
```

- `num` is assigned the value 7.
- The `if` condition checks if `num` modulo 2 is equal to 0 (even). Since 7 % 2 is not equal to 0, it moves to the `else` block.

- The `else` block is executed, printing "The number is odd."

Example 3: Checking Eligibility for Voting

```
let age = 17;

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

- `age` is assigned the value 17.
- The `if` condition checks if `age` is greater than or equal to 18 (voting age). Since 17 is less than 18, it moves to the `else` block.
- The `else` block is executed, printing "You are not eligible to vote yet."

`if-else if-else` Statement

- An `if-else if-else` statement allows you to perform different actions based on multiple conditions.
- If none of the conditions in an `if-else if-else` chain are true, the code inside the final `else` block will execute.

Example 1: Grading System

```
let score = 85;

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

- `score` is assigned the value 85.
- The `if` condition checks if `score` is greater than or equal to 90 (A grade). Since 85 is not greater than or equal to 90, it moves to the first `else if` condition.
- The first `else if` condition checks if `score` is greater than or equal to 80 (B grade). Since 85 is greater than 80, it prints "B."
- If neither of the above conditions is met, the `else` block is executed, printing "C or below."

Example 2: Time of the Day Greeting

```
let currentHour = 18;
```

```
if (currentHour < 12) {
  console.log("Good morning!");
} else if (currentHour < 18) {
  console.log("Good afternoon!");
} else {
  console.log("Good evening!");
}
```

- `currentHour` is assigned the value 18.
- The first `if` condition checks if `currentHour` is less than 12 (morning). Since 18 is not less than 12, it moves to the second `else if` condition.
- The second `else if` condition checks if `currentHour` is less than 18 (afternoon). Since 18 is equal to 18, it prints "Good evening."
- If neither of the above conditions is met, the `else` block is executed.

Example 3: Classifying Triangles

```
let sideA = 3;
let sideB = 4;
let sideC = 5;

if (sideA === sideB && sideB === sideC) {
  console.log("Equilateral triangle");
} else if (sideA === sideB || sideB === sideC || sideA === sideC) {
  console.log("Isosceles triangle");
} else {
  console.log("Scalene triangle");
}
```

- Three sides of a triangle are given (3, 4, 5).
- The first `if` condition checks if all sides are equal, printing "Equilateral triangle" if true.
- The second `else if` condition checks if at least two sides are equal (Isosceles), printing "Isosceles triangle" if true.
- If neither condition is met, it prints "Scalene triangle."

Nested `if` Statements

Nested `if` statements allow for more complex decision-making by allowing multiple conditions to be checked in sequence.
Each nested `if` statement

- has its own set of parentheses enclosing the conditional expression and any associated code blocks.
- can also include additional `else if` statements to further refine the logic flow.
- can have an optional `else` clause that executes when none of the previous conditions are satisfied.

Example 1: Checking Number Parity

```
```

```
let numberToCheck = 15;

if (numberToCheck > 0) {
  if (numberToCheck % 2 === 0) {
    console.log("The number is a positive even number.");
  } else {
    console.log("The number is a positive odd number.");
  }
} else {
  console.log("The number is not positive.");
}
```

- `numberToCheck` is assigned the value 15.
- The outer `if` condition checks if `numberToCheck` is positive. Since 15 is positive, it proceeds to the inner `if` condition.
- The inner `if` condition checks if `numberToCheck` modulo 2 is equal to 0 (even). Since 15 % 2 is not equal to 0, it prints "The number is a positive odd number."

Example 2: Ticket Price Based on Age

```
let passengerAge = 25;

if (passengerAge < 18) {
  console.log("Child ticket: $10");
} else {
  if (passengerAge < 60) {
    console.log("Adult ticket: $20");
  } else {
    console.log("Senior citizen ticket: $15");
  }
}
```

- `passengerAge` is assigned the value 25.
- The outer `if` condition checks if `passengerAge` is less than 18 (child). Since 25 is not less than 18, it moves to the inner `if` condition.
- The inner `if` condition checks if `passengerAge` is less than 60 (adult). Since 25 is less than 60, it prints "Adult ticket: $20."

Example 3: Validating User Input

```
let userInput = "admin";
let password = "1234";

if (userInput === "admin") {
  if (password === "1234") {
    console.log("Login successful!");
  } else {
```

```
    console.log("Incorrect password.");
  }
} else {
  console.log("Invalid username.");
}
```

- `userInput` is "admin," and `password` is "1234."
- The outer `if` condition checks if `userInput` is "admin." If true,
- it moves to the inner `if` condition.
- The inner `if` condition checks if `password` is "1234." Since it is, it prints "Login successful!"

Why `if` Statements Matter ?

- `if` statements are essential for creating dynamic and responsive programs.
- They allow you to control the flow of your code based on conditions, enabling different paths for different scenarios.
- Understanding and mastering `if` statements is foundational to writing effective JavaScript code.

This detailed explanation and examples should help you grasp the concepts of `if` statements in JavaScript and how to use them effectively.