Continue

If then else statement example

Conditional statements are used to perform different actions based on different conditions. Conditional Statements Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this. In JavaScript we have the following conditional statements: Use if to specify a block of code to be executed, if a specified condition is true Use else to specify a block of code to be executed, if the same condition is false Use else if to specify a new condition to test, if the first condition is false Use switch to specify many alternative blocks of code to be executed The switch statement is described in the next chapter. The if Statement Use the if statement to specify a block of JavaScript code to be executed if a condition is true. Syntax if (condition) { // block of code to be executed if the condition is true } Note that if is in lowercase letters. Uppercase letters (If or IF) will generate a JavaScript error. Make a "Good day" greeting if the hour is less than 18:00: if (hour < 18) { greeting = "Good day"; } The result of greeting will be: Try it Yourself » Use the else statement to specify a block of code to be executed if the condition is false. if (condition) { // block of code to be executed if the condition is true } else { // block of code to be executed if the condition is false } If the hour is less than 18, create a "Good day" greeting, otherwise "Good evening": if (hour < 18) { greeting = "Good day"; } else { greeting = "Good evening"; } The result of greeting will be: Try it Yourself » The else if Statement Use the else if statement to specify a new condition if the first condition is false. Syntax if (condition1) { // block of code to be executed if condition1 is true } else if (condition2) { // block of code to be executed if the condition1 is false and condition2 is true } else { // block of code to be executed if the condition1 is false and condition2 is false } If time is less than 10:00, create a "Good morning" greeting, if not, but time is less than 20:00, create a "Good day" greeting, otherwise a "Good evening": if (time < 10) { greeting = "Good morning"; } else if (time < 20) { greeting = "Good day"; } else { greeting = "Good evening"; } The result of greeting will be: Try it Yourself » This example will write a link to either W3Schools or to the World Wildlife Foundation (WWF). By using a random number, there is a 50% chance for each of the links. let text; if (Math.random() < 0.5) { text = "Visit W3Schools"; } else { text = "Visit WWF"; } document.getElementById("demo").innerHTML = text; Try it Yourself » Kenneth Leroy Busbee The if–then–else construct, sometimes called if-then, is a two-way selection structure common across many programming languages. Although the syntax varies from language to language, the basic structure looks like: If (boolean condition) Then (consequent) Else (alternative) End If Discussion We are going to introduce the control structure from the selection category that is available in every high level language. It is called the if then else structure. Asking a question that has a true or false answer controls the if then else structure. It looks like this: if the answer to the question is true then do this else because it is false do this In most languages, the question (called a test expression) is a Boolean expression. The Boolean data type has two values – true and false. Let’s rewrite the structure to consider this: if expression is true then do this else because it is false do this Some languages use reserved words of: “if”, “then” and “else”. Many eliminate the “then”. Additionally the “do this” can be tied to true and false. You might see it as: if expression is true action true else action false And most languages infer the “is true” you might see it as: if expression action true else action false The above four forms of the control structure are saying the same thing. The else word is often not used in our English speaking today. However, consider the following conversation between a mother and her child. Child asks, “Mommy, may I go out side and play?” Mother answers, “If your room is clean then you may go outside and play or else you may go sit on a chair for five minutes as punishment for asking me the question when you knew your room was dirty.” Let’s note that all of the elements are present to determine the action (or flow) that the child will be doing. Because the question (your room is clean) has only two possible answers (true or false) the actions are mutually exclusive. Either the child 1) goes outside and plays or 2) sits on a chair for five minutes. One of the actions is executed; never both of the actions. One Choice – Implied Two-Way Selection Often the programmer will want to do something only if the expression is true, that is with no false action. The lack of a false action is also referred to as a “null else” and would be written as: if expression action true else do nothing Because the “else do nothing” is implied, it is usually written in short form like: if expression action true Key Terms if then else A two-way selection control structure. mutually exclusive Items that do not overlap. Example: true or false. References Conditional statements in programming are used to control the flow of a program based on certain conditions. These statements allow the execution of different code blocks depending on whether a specified condition evaluates to true or false, providing a fundamental mechanism for decision-making in algorithms. In this article, we will learn about the basics of Conditional Statements along with their different types.What are Conditional Statements in Programming?Conditional statements in Programming, also known as decision-making statements, allow a program to perform different actions based on whether a certain condition is true or false. They form the backbone of most programming languages, enabling the creation of complex, dynamic programs.5 Types of Conditional Statements in ProgrammingConditional statements in programming allow the execution of different pieces of code based on whether certain conditions are true or false. Here are five common types of conditional statements:5 Types of Conditional Statements in Programming1. If Conditional Statement:The if statement is the most basic form of conditional statement. It checks if a condition is true. If it is, the program executes a block of code.Syntax of If Conditional Statement:if (condition) { // code to execute if condition is true}if condition is true, the if code block executes. If false, the execution moves to the next block to check.Use Cases of If Conditional Statement:Checking a single condition and executing code based on its result.Performing actions based on user input.Applications of If Conditional Statement:Validating user inputs.Basic decision-making in algorithms.Advantages of If Conditional Statement:Simple and straightforward.Useful for handling basic decision logic.Disadvantages of If Conditional Statement:Limited to checking only one condition at a time.Not suitable for complex decision-making.Implementation of If Conditional Statement: C++ #include using namespace std; int main() { int x = 10; if (x > 0) { cout 0: print("x is positive") # Print a message if x is positive if __name__ == "__main__": main() C# using System; class Program { static void Main(string[] args) { int x = 10; if (x > 0) { Console.WriteLine("x is positive"); } // This code checks if the variable x is positive. } } JavaScript function main() { let x = 10; // Check if x is greater than 0 if (x > 0) { console.log("x is positive"); // Print a message if x is positive } } // Call the main function main(); 2. If-Else Conditional Statement:The if-else statement extends the if statement by adding an else clause. If the condition is false, the program executes the code in the else block. Syntax of If-Else Conditional Statement:if (condition) { // code to execute if condition is true} else { // code to execute if condition is false}if condition is true, the if code block executes. If false, the execution moves to the else block.Use Cases of If-Else Conditional Statement:Executing one block of code if a condition is true and another block if it's false.Handling binary decisions.Applications of If-Else Conditional Statement:Error handling: For example, displaying an error message if user input is invalid.Program flow control: Directing program execution based on conditions.Advantages of If-Else Conditional Statement:Handles binary decisions efficiently.Clear and concise syntax.Disadvantages of If-Else Conditional Statement:Limited to binary decisions.May become verbose in complex scenarios.Implementation of If-Else Conditional Statement: C++ #include using namespace std; int main() { int x = -10; if (x > 0) { cout 0: print("x is positive") else: print("x is not positive") # Call the main function to execute the code if __name__ == "__main__": main() C# using System; class Program { static void Main(string[] args) { int x = -10; // Check if x is greater than 0 if (x > 0) { Console.WriteLine("x is positive"); } else { Console.WriteLine("x is not positive"); } } } JavaScript // Main function function main() { // Define the value of x const x = -10; // Check if x is greater than 0 if (x > 0) { console.log("x is positive"); } else { console.log("x is not positive"); } } // Call the main function to execute the program main(); 3. if-Else if Conditional Statement:The if-else if statement allows for multiple conditions to be checked in sequence. If the if condition is false, the program checks the next else if condition, and so on. Syntax of If-Else if Conditional Statement:if (condition1) { // code to execute if condition1 is true} else if (condition2) { // code to execute if condition2 is true} else { // code to execute if all conditions are false}In else if statements, the conditions are checked from the top-down, if the first block returns true, the second and the third blocks will not be checked, but if the first if block returns false, the second block will be checked. This checking continues until a block returns a true outcome.Use Cases of If-Elif-Else Conditional Statement:Handling multiple conditions sequentially.Implementing multi-way decision logic.Applications of If-Elif-Else Conditional Statement:Implementing menu selection logic.Categorizing data based on multiple criteria.Advantages of If-Elif-Else Conditional Statement:Allows handling multiple conditions in a structured manner.Reduces the need for nested if-else statements.Disadvantages of If-Elif-Else Conditional Statement:Can become lengthy and harder to maintain with many conditions.The order of conditions matters; incorrect ordering can lead to unexpected behavior.If-Else if Conditional Statement Implementation: C++ #include using namespace std; int main() { int x = 0; if (x > 0) { cout 0) { console.log("x is positive"); } // If not positive, check if the number is negative else if (x < 0) { console.log("x is not positive"); } // If neither positive nor negative // the number is zero else { console.log("x is not zero"); } 4. Switch Conditional Statement:The switch statement is used when you need to check a variable against a series of values. It’s often used as a more readable alternative to a long if-else if chain. In switch expressions, each block is terminated by a break keyword. The statements in switch are expressed with cases.Switch Conditional Statement Syntax:switch (variable) { case value1: // code to execute if variable equals value1 break; case value2: // code to execute if variable equals value2 break; default: // code to execute if variable doesn't match any value}Use Cases of Switch Statement:Selecting one of many code blocks to execute based on the value of a variable.Handling multiple cases efficiently.Applications of Switch Statement:Processing user choices in a menu.Implementing state machines.Advantages of Switch Statement:Provides a clean and efficient way to handle multiple cases.Improves code readability when dealing with many conditions.Disadvantages of Switch Statement:Limited to equality comparisons, cannot use range checks or complex conditions.Lack of fall-through control can lead to unintentional bugs if not used carefully.Switch Conditional Statement Implementation: C++ #include ; using namespace std; int main() { int x = 2; switch (x) { case 1: cout 0 ? "x is positive" : "x is not positive"; // Print the result to the console console.log(result); Difference between Types of Conditional Statements in Programming:Conditional StatementPurposeUsageExampleifExecute code if condition is trueSingle conditionif x > 5: print("x is greater than 5")if-elseExecute one block if condition is true, another if falseTwo mutually exclusive possibilitiesif x > 5: print("x is greater than 5") else: print("x is not greater than 5")if-elif-elseExecute based on multiple conditionsMultiple conditions, sequential evaluationpython if x > 5: print("x is greater than 5") elif x == 5: print("x is equal to 5") else: print("x is less than 5")switch-caseSelect one of many code blocks to execute based on a variableMatching variable against multiple casesjava switch (day) { case 1: System.out.println("Monday"); break; case 2: System.out.println("Tuesday"); break; default: System.out.println("Unknown day"); }Difference between If Else and Switch Case:Featureif-else Statementswitch StatementMultiple ConditionsSupports multiple conditions using else ifSupports multiple cases using case statementsEquality ComparisonCan handle complex conditions with relational operatorsTypically checks equality with case valuesRange ComparisonCan handle ranges using logical operatorsTypically handles discrete values, not suitable for rangesFall-ThroughExecutes the first true condition and exitsContinues executing cases until break or end.Default CaseOptional else block for default behaviordefault case for unmatched valuesExpression TypeSupports any boolean expression in the conditionTypically used with expressions resulting in discrete valuesReadability and MaintainabilityReadability may decrease with nested conditionsReadability can be maintained for multiple casesUse CasesSuitable for various conditions and complex logicSuitable for scenarios with distinct, known valuesBest Practices for Conditional Statements in Programming:Keep it simple: Avoid complex conditions that are hard to understand. Break them down into simpler parts if necessary.Use meaningful names: Your variable and function names should make it clear what conditions you’re checking.Avoid deep nesting: Deeply nested conditional statements can be hard to read and understand. Consider using early returns or breaking your code into smaller functions.Comment your code: Explain what your conditions are checking and why. This can be especially helpful for complex conditions.In conclusion, Conditional statements are a fundamental part of programming, allowing for dynamic and interactive programs. By understanding and using them effectively, you can create programs that are more efficient, readable, and maintainable. An if else statement in programming is a basic programming technique that allows you to make decisions based on certain conditions. It allows your program to execute different pieces of code depending on whether the specified condition evaluates to true or false. This capability is crucial in building dynamic and functional applications. Importance of If Else Statement:The importance of if else statements lies in their ability to control the execution of a program. Using if else statements allows developers to apply logic that responds to situations, making programs more versatile and powerful. Whether manipulating user statements, manipulating data, or controlling program flow, if else statements play an important role in programming. Basic Syntax of If Else Statement:Generally, the basic syntax of if felse statements follows this pattern: if (condition) { // Code block to execute if condition is true } else { // Code block to execute if condition is false } In this syntax: The `if` keyword begins a conditional statement.The condition is enclosed in parentheses `()`.If the condition evaluates to true, the code block immediately following the `if` statement is executed.If the condition evaluates to false, the code block is executed in the `else` statement.If Else Statement in C:Here are the implementation of if else statement in C language: C #include int main() { // Declare and initialize the variable num int num = 10; // Check if num is greater than 0 if (num > 0) { // If num is greater than 0, print "Number is positive." printf("Number is positive."); } else { // If num is not greater than 0, print "Number is non-positive." printf("Number is non-positive."); } return 0; } OutputNumber is positive.If Else Statement in C++:Here are the implementation of if else statement in C++ language: C++ #include using namespace std; int main() { // Declare and initialize the variable num int num = 10; // Check if num is greater than 0 if (num > 0) { // If num is greater than 0, print "Number is positive." cout 0) { // If num is greater than 0, print "Number is positive." console.log("Number is positive."); } else { // If num is not greater than 0, print "Number is non-positive." console.log("Number is non-positive."); } OutputNumber is positive.This if else statement checks if the variable `num` is greater than 0. If so, the function says "Numbers are good."; Otherwise, it prints "Number is non-positive". ConclusionIf else statements are necessary to implement conditional logic in policy. They provide tools to monitor the system based on specific scenarios, enabling developers to create dynamic and functional applications. Understanding how to properly use if else statements is key to mastering programming languages like C++, Java, etc.