Conditional Statements - if-else if-else; Switch cases; Loops; Intro to Functions

Relevel
by Unacademy

If statement

If statement - a simple decision-making statement used to decide whether a particular statement or block of statements will execute or not. In that case, i.e., if a certain condition is true, then a block of statement is executed; otherwise, not.

Syntax:

```
if(condition)
{
   // Statements to execute if
   // condition is true
}
```

If-else statement

If-else: The if statement alone tells us that if a condition is true, it will execute a block of statements, and if the condition is false, it won't. But what if we want to do something else if the condition is false. Here comes the else statement. We can use the else statement with the if statement to execute a code block when the condition is false.

Syntax:

```
if (condition)
// Execute if condition is true
else
        // Execute if condition is false
```

Example

Odd-even check:

```
if(num % 2 === 0)
        console.log("even")
else
        console.log("odd")
```

Nested-if statement

A nested if is an if statement which targets another if or else. Nested if statement means an if statement inside an if statement. Yes, Javascript allows us to use nest if statements within if statements.

Nested-if statement

Syntax:

```
if (condition1)
{
   // Executes when condition1 is true
   if (condition2)
   {
      // Executes when condition2 is true
   }
}
```

Example:

```
function f1() {
if(num % 2 === 0)
          if(num === 4)
                    console.log("hello")
}
```

If-else-if ladder

In this, the user decides amongst multiple options. The if statements are executed from the top down. When one of the conditions controlling 'if' is true, the statement associated with it is executed and bypasses the rest of the ladder. If none of the conditions are true, then it executes the final else statement.

If-else-if ladder

Syntax:

```
if (condition)
    statement;
else if (condition)
    statement;
.
.
else
    statement;
```

Example:

```
function f1() {
if(num % 2 === 0)
console.log("hello")
else if(num == 3)
          console.log("hi")
}
```

Switch case statement

The switch statement is a multiway branch statement. It provides an easy way to dispatch execution to different parts of the code based on the value of the expression.

Switch case statement

Syntax:

```
switch(expression) {
  case n:
          code block
    break;
  case n:
          code block
    break;
  default:
    default code block
}
```

Switch case statement

- An expression can be type byte, short, int char, or enumeration.
- Duplicate case values are not allowed.
- A default statement is optional.
- The break statement - used inside the switch to terminate the sequence of statements

Syntax:

```
switch(1) {
  case 1:console.log("hi");break;
  case 2:console.log("hi2");break;
  default:console.log("hello");break;
}
```

Comparison between if-else and switch

- if-else statement uses multiple statements for multiple choices. switch statement uses a single expression for multiple choices.
- Either if statement will be executed or else statement is executed. The switch statement executes one case after another till a break statement appears or the end of the switch statement is reached.
- Switch cases only test equality whereas if-else if-else can test all types of conditions.

Loops

For Loops:

For loops are also called entry-controlled loops because in for loop, the condition is checked before entering in the loop body. If the required condition is met i.e if the expression for condition checking results out to be an actual value, then the statements written inside the loop body executes. If the condition evaluates to false the control will not enter the loop body.

Example Problem Statement

Consider you want to print the first 5 natural numbers

```
for(let i = 0;i < 5;i++)
        console.log(i);
```

While Loops

While loops are also called entry loops because the condition is checked before the loop control gets into the loop body.

Problem Statement

Consider you want to print the first 5 natural numbers

```
while(i < 5) {
        console.log(i);
        i++;
}
```

Do-While Loops

- Do while loops are also called exit controlled loops because the condition is checked after the loop body executes. So, if the condition expression evaluates to false for the first iteration the do-while loop will at least execute once.
- Do-while loops can be used if the number of times we need to execute the loop is not known before, and the loop should execute at least once.

Problem Statement

Consider you want to print the first 5 natural numbers

```
do {
        console.log(i);
        i++;
} while(i < 5);
```

Nested Loops

Loops inside loops. It's the way to declare one loop inside another loop using any of the above loops.

Problem Statement

Consider you want to print the first 5 natural numbers

```
for(let i = 0;i < 5;i++)
          for(let j = 0;j < 5;j++)
                    console.log(j);
```

Intro to functions

A JavaScript function is a block of code designed to perform a particular task. It is executed when "something" invokes it (calls it).

Syntax

```
function name(parameter1, parameter2,
parameter3) {
  // code to be executed
}
```

Arrow function

A JavaScript function is a block of code designed to perform a particular task. It is executed when "something" invokes it (calls it).

Syntax

```
sum = (a1, a2) => {
  return a1 + a2;   // The function returns the sum of a1 and a2
}
let x = sum(5, 6);
```

This works exactly the same way as the originally declared function.

Arrow function

A JavaScript function is a block of code designed to perform a particular task. It is executed when "something" invokes it (calls it).

Syntax

```
sum = (a1, a2) => {
  return a1 + a2;   // The function returns the sum of a1 and a2
}
let x = sum(5, 6);
```

This works exactly the same way as the originally declared function.

Assigning functions to a variable

This is a function expression and so only defined when that line of the code is reached.

Syntax

```
let x = function sum(a1, a2){
  return a1 + a2;   // The function returns the sum of a1 and a2
}
```

Assignment

- Program to check if numbers are palindrome numbers or not (A number is said to be palindrome if remains the same when it is reversed Eg: 121,454 are palindrome whereas 345,5898 are not palindrome)

- Program to check prime numbers. A number is prime if it has two factors only(1 and itself). Examples - 2,3,5,7,11,13.

Upcoming Class

- Let’s solve some problems using conditional statements and loops

Thank you!