
# Coding Bootcamp


# JavaScript

# Control Flows

# Learning Objectives

By the end of this lesson, you will be able to:

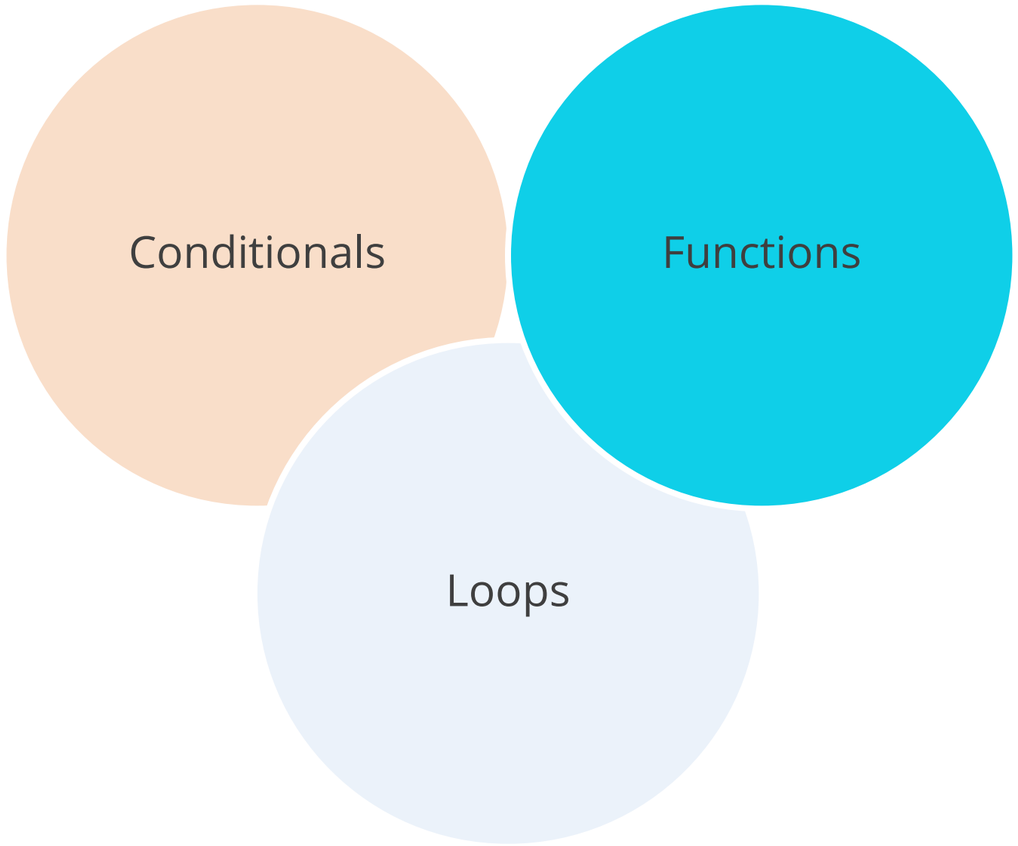
- List the categories of control flow statements in JavaScript for analyzing program execution flow
- Describe syntaxes for different conditional statements in JavaScript for accurate decision-making in code
- Implement various looping constructs to manage repetitive tasks for efficient code execution
- Use break and continue statements to control loop flow for optimizing loop performance

TECHNOLOGY

# Control Flows and Its Types

# Control Flow Statements

They are used to control the flow of execution in a program. The basic types of control flow statements are:



# Conditionals

Conditionals control the flow of a program based on whether a condition is true or false. The primary conditional statements are:

- if statement
- if...else statement
- else if statement:
- switch statement

# If Statement

Executes a block of code if a specified condition is true

**Syntax**

```
if (condition) {
  // code to be executed if the
condition is true
}
```

**Example**

```
let age = 18;
if (age >= 18) {
  console.log('You are an adult.');
}
```

# If Else Statement

Executes one block of code if a condition is true, and another block of code if the condition is false

## Syntax

```
if (condition) {
  // code to be executed if the
condition is true
} else {
  // code to be executed if the
condition is false
}
```

## Example

```
let age = 16;
if (age >= 18) {
  console.log('You are an adult.');
} else {
  console.log('You are a minor.');
}
```

# Else If Statement

Specifies a new condition to test if the first condition is false

## Syntax

```
if (condition1) {
  // code to be executed if condition1
is true
} else if (condition2) {
  // code to be executed if condition2
is true
} else {
  // code to be executed if both
conditions are false
}
```

## Example

```
let score = 75;
if (score >= 90) {
  console.log('Grade: A');
} else if (score >= 80) {
  console.log('Grade: B');
} else if (score >= 70) {
  console.log('Grade: C');
} else {
  console.log('Grade: F');
}
```

# Switch Statement

Evaluates an expression, matches its value to a corresponding case clause, and executes the statements associated with that case

## Syntax

```
switch (expression) {
  case value1:
    // code to be executed if
expression equals value1
    break;
  case value2:
    // code to be executed if
expression equals value2
    break;
  default:
    // code to be executed if
expression doesn't match any case
}
```

## Example

```
let fruit = 'apple';
switch (fruit) {
  case 'apple':
    console.log('Apples are red.');
    break;
  case 'banana':
    console.log('Bananas are yellow.');
    break;
  default:
    console.log('Unknown fruit.');
}
```

# Implementing if-else Conditions for Applying Promo Codes

**Problem Statement:**

You have been asked to demonstrate how to implement if-else conditions to apply promo codes and calculate discounts based on specific conditions

**Outcome:**

By completing this task, you demonstrated how to implement if-else conditions to apply promo codes and calculate discounts based on specific conditions. You wrote an algorithm to handle various promo codes and amounts, and then tested the code with different scenarios to ensure its correctness and effectiveness.

**Note**: Refer to the demo document for detailed steps:
01_Implementing_if-else_Conditions_for_Applying_Promo_Codes

ASSISTED PRACTICE

# Assisted Practice: Guidelines

**Steps to be followed are:**

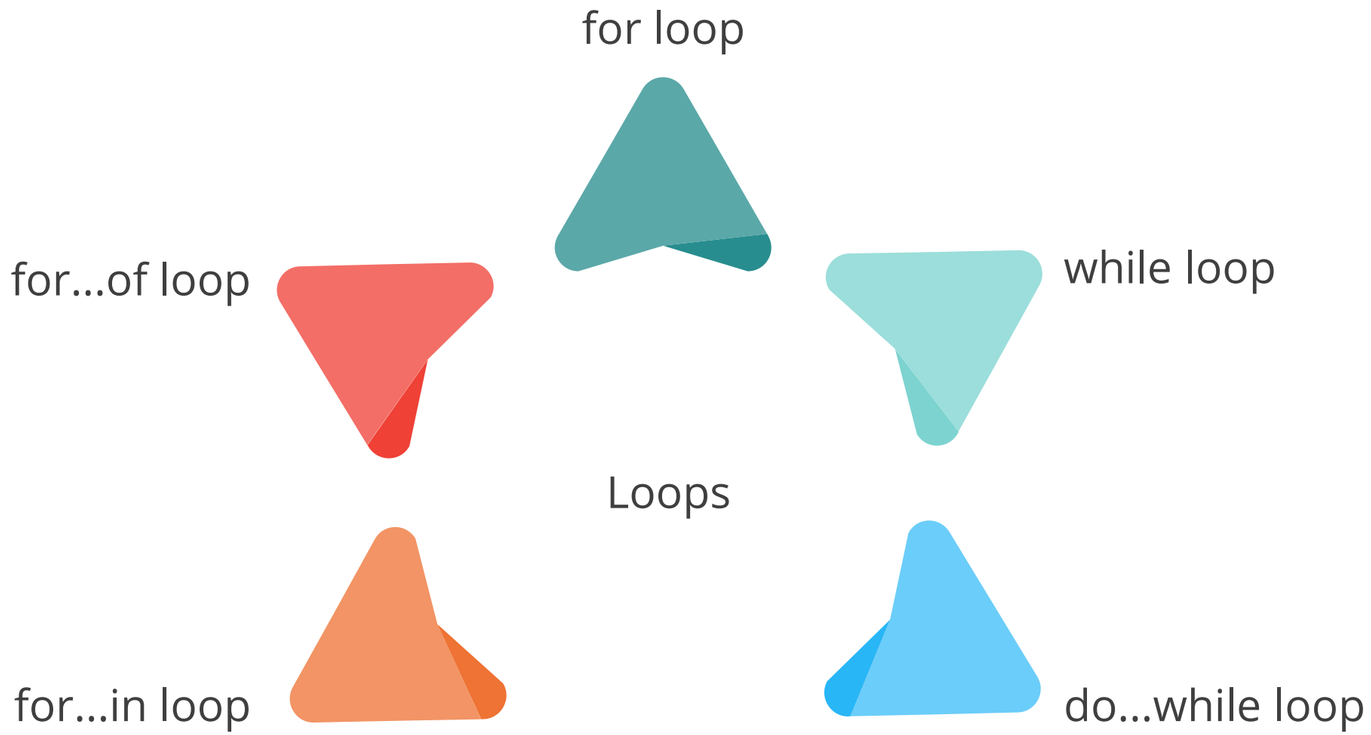
1. Write an algorithm to apply promo codes and calculate discounts
2. Test the code with different promo codes and amounts

TECHNOLOGY

# Loops and Its Types

# Loops

They execute a block of code multiple times, depending on a given condition. The main types of loops are:



# For Loop

Repeats a block of code a specified number of times

**Syntax**

```
for (initialization; condition;
increment) {
  // code to be executed
}
```

**Example**

```
for (let i = 0; i < 5; i++) {
  console.log('Iteration:', i);
}
```

# While Loop

Repeats a block of code as long as a specified condition is true

**Syntax**

```
while (condition) {
  // code to be executed
}
```

**Example**

```
let count = 0;
while (count < 5) {
  console.log('Count:', count);
  count++;
}
```

# do...while Loop

Executes the block of code once before checking the condition

**Syntax**

```
do {
  // code to be executed
} while (condition);
```

**Example**

```
let count = 0;
do {
  console.log('Count:', count);
  count++;
} while (count < 5);
```

# for...in Loop

Iterates over the properties of an object

**Syntax**

```
for (variable in object) {
  // code to be executed
}
```

**Example**

```
let count = 0;
do {
  console.log('Count:', count);
  count++;
} while (count < 5);
```

# for...of Loop

Iterates over iterable objects such as arrays or strings

**Syntax**

```
for (variable of iterable) {
  // code to be executed
}
```

**Example**

```
let count = 0;
do {
  console.log('Count:', count);
  count++;
} while (count < 5);
```

# Break and Continue

They are used to control the flow of loops.

**break**

```
for (let i = 0; i < 10; i++) {
  if (i === 5) {
    break;
  }
  console.log('Iteration:', i);
}
```

- The **break** statement is used to terminate the loop or switch statement prematurely.
- When a **break** statement is encountered, the control is immediately transferred to the statement following the terminated loop or switch.

# Break and Continue

**continue**

```
for (let i = 0; i < 10; i++) {
  if (i % 2 === 0) {
    continue;
  }
  console.log('Iteration:', i);
}
```

- The **continue** statement is used to skip the current iteration of the loop and proceed to the next iteration.
- When a **continue** statement is encountered, the remaining statements in the current iteration are skipped, and the loop proceeds with the next iteration.

TECHNOLOGY

# Functions



simplilearn

# What are Functions?

They are blocks of code designed to perform a particular task and are executed when called from elsewhere in the code. Different ways to define a function are:

- Function declaration
- Function expression
- Arrow functions

# Function Declaration

Defines a function with a given name

**Syntax**

```
function functionName(parameters) {
  // code to be executed
}
```

**Example**

```
// Example
function greet(name) {
  return `Hello, ${name}!`;
}

console.log(greet('Alice'));
// Output: Hello, Alice!
```

# Function Expression

Defines a function as part of a larger expression, often an assignment

**Syntax**

```
const functionName = function(parameters) {
  // code to be executed
};
```

**Example**

```
// Example
const greet = function(name) {
  return `Hello, ${name}!`;
};

console.log(greet('Bob'));
// Output: Hello, Bob!
```

# Arrow Functions

Provides more concise syntax for writing function expressions

**Syntax**

```
const functionName = (parameters) => {
  // code to be executed
};
```

**Example**

```
// Example
const greet = (name) => `Hello, ${name}!`;

console.log(greet('Charlie'));
// Output: Hello, Charlie!
```

# Key Takeaways

- Control flow statements are used to control the flow of execution in a program.
- The switch case statement evaluates and executes the expression after matching its value to a case clause.
- The for...in loop in JavaScript allows one to iterate over the properties of an object.
- The continue statement is used to skip the current iteration of the loop and proceed to the next iteration.

TECHNOLOGY

# Thank You



simplilearn