Automation Testing

Introduction to JavaScript

simplilearn

A Day in the Life of an Automation Test Engineer

Anna is working in an organization as an Automation Testing Engineer.

She has been asked to develop a website to handle user inputs and validation for which she will be working with the JavaScript codes.

To achieve all of the above with some additional features, she will learn the basics of JavaScript, including variables, keywords, data types, control statements, functions, and arrays which will help her to find the solution for the given scenario.

Learning Objectives

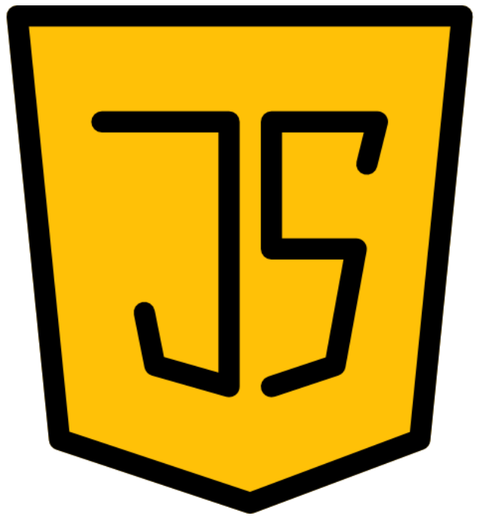
By the end of this lesson, you will be able to:

- Define JavaScript and its functions
- Describe and use variables, arrays, conditionals, and loops in JavaScript programming
- Describe arithmetic operators
- Differentiate between conditional and iterative statements

JavaScript: Overview

Introduction to JavaScript

JavaScript is a lightweight scripting programming language used to make web pages interactive. It can also calculate, validate, and manipulate the data.



In addition, it can insert dynamic text into HTML and CSS.

Why JavaScript?

The four most important features of JavaScript are:

01

Simple tasks are carried out in a straightforward manner in JavaScript.

02

JavaScript is completely compatible with HTML and CSS.

03

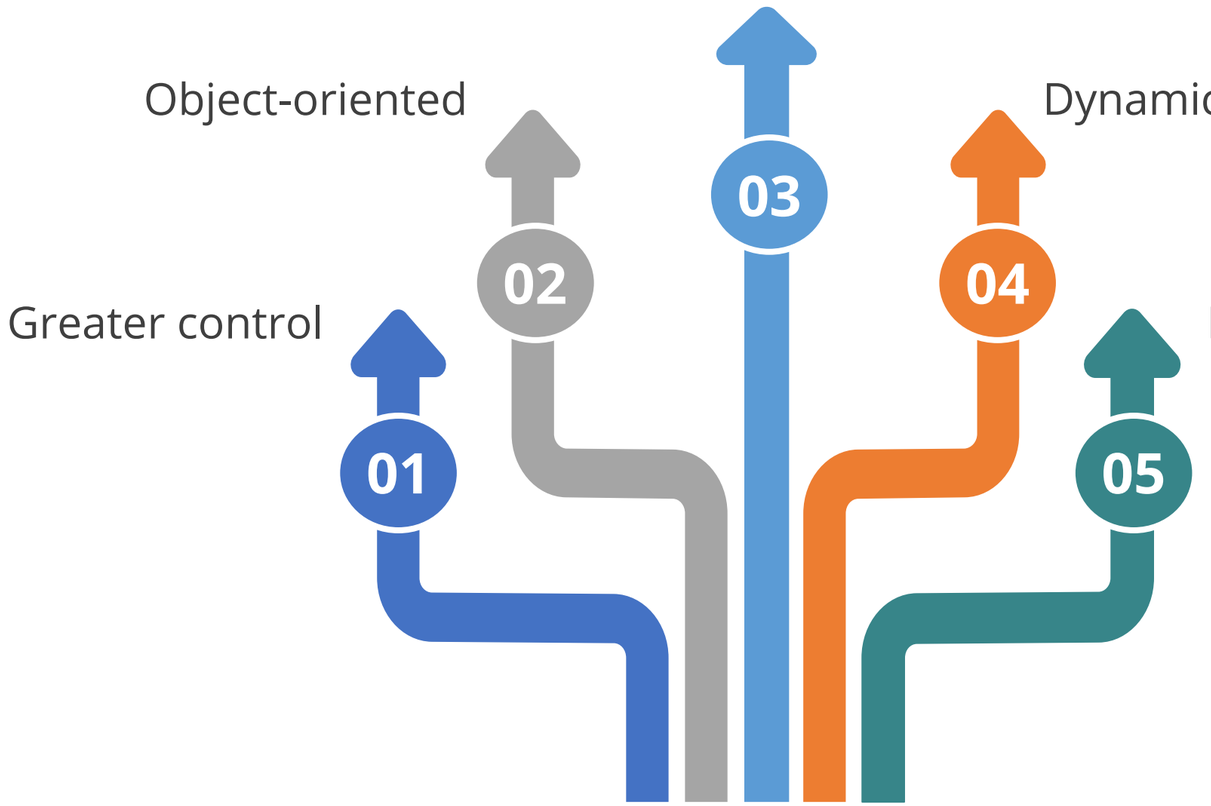
React, Angular, and other JavaScript frameworks are widely used to create seamless web and mobile apps.

04

All major browsers support it, and most of them enable it by default.

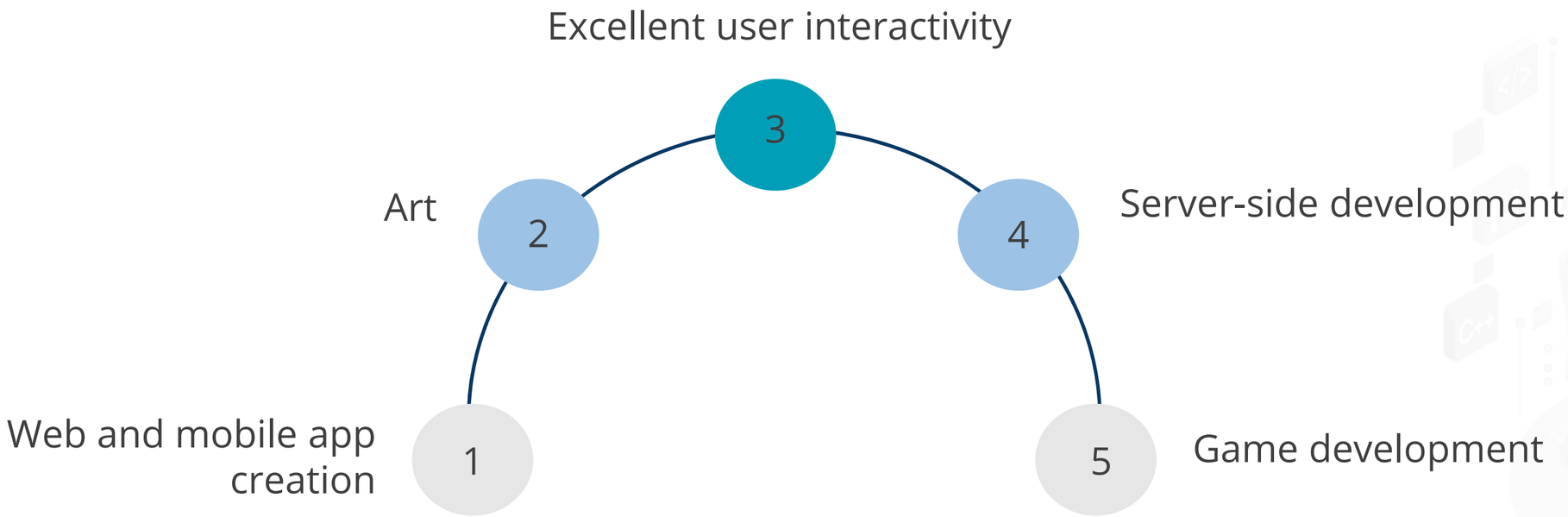
Features of JavaScript

The various features of JavaScript are:



JavaScript Application

JavaScript is used to create interactive websites. It is mainly used for:



Write and Execute Javascript Program Using Node

Problem Statement:

You are asked to write and execute JavaScript program using Node.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to write and execute JavaScript program using Node are:

1. Write and execute JavaScript program

JavaScript Programming Concepts

Variables

Variables are the names users give to the memory locations in a computer program where values are stored.

There are two types of variables in JavaScript:

Local variable

A local variable is declared inside block or function. It is accessible within the function or block only.

Global variable

A global variable is accessible from any function. A variable declared outside the function or declared with window object.

Variables

The following are some rules for declaring a variable (also known as an identifiers):

- A variable can have alphabets, digits, and underscore.
- A variable name can start with any letter of the alphabet or underscore only.
- A variable can’t start with a digit. And no whitespace is allowed within the variable name.
- A variable name must not be a reserved word or keyword.

Data Types

A data type is a classification tells the compiler or interpreter how the programmer plans to use the data.

Integer, real, character or string, and boolean data are all supported by most programming languages.

Data Types

There are two types of data types in:

Primitive data type

Primitive data types are predefined types in the programming language.

Non-Primitive data type

Non-primitive data types are reference types that refer to objects.

Primitive Data Types

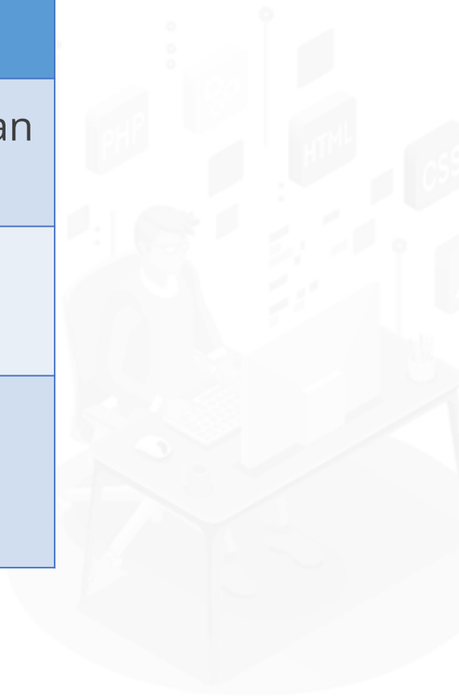
There are five types of primitive data types in JavaScript. They are as follows:

Data Type	Description
String	Represents sequence of characters
Number	Represents numeric values
Boolean	Represents the boolean value of either false or true
Undefined	Represents undefined value
Null	Represents null

Non-Primitive Data Types

The non-primitive data types are as follows:

Data Type	Description
Object	Represents an instance through which users can access members.
Array	Represents a group of similar values
RegExp	Represents regular expression

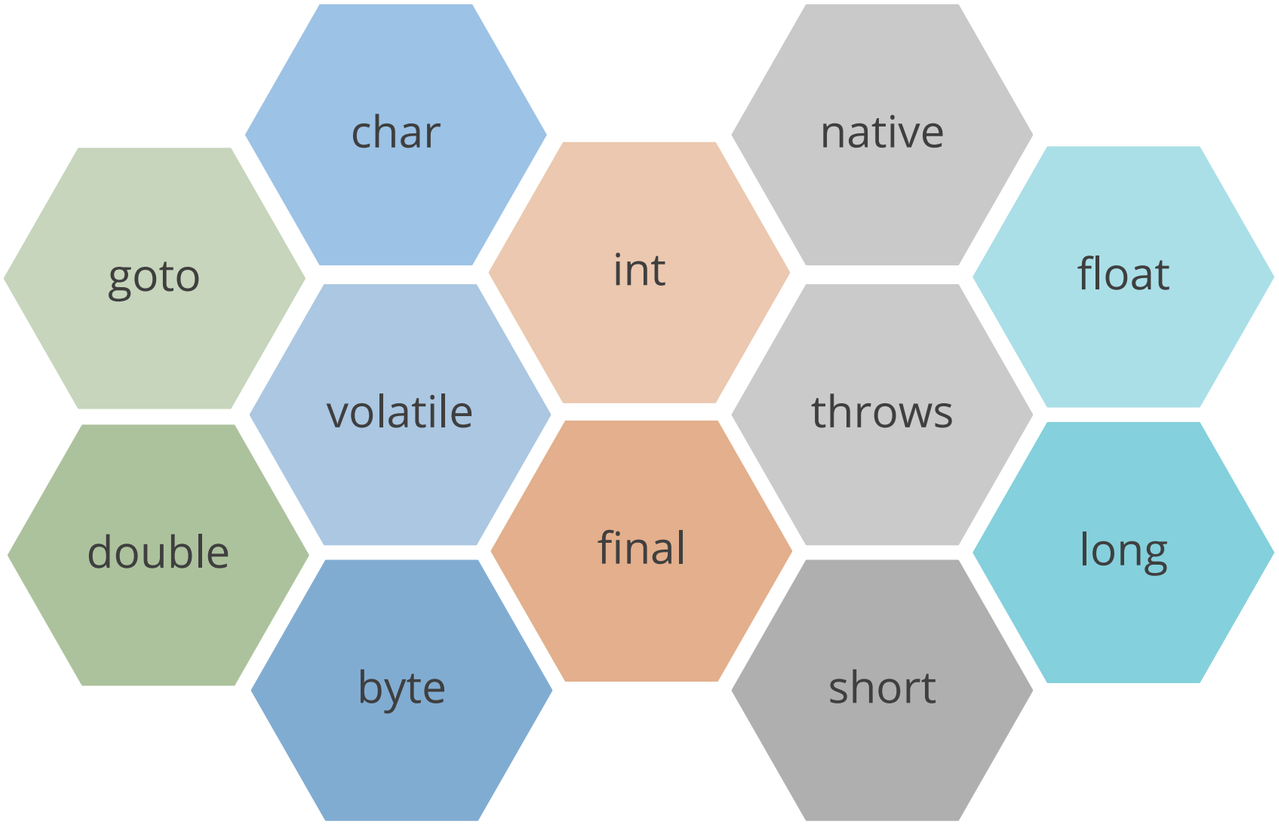


Keyword

Keywords are reserved words in programming that have a specific meaning for the compiler.

Keyword

The lists of some standard reserved words are given below:



String

A string is a collection of one or more characters, which can be letters, numbers, or symbols.

There are two ways to create a string in JavaScript:

- By string literal
- By string object

String Methods

Methods	Description
charAt()	It returns the character value at the specified index.
charCodeAt()	It returns the Unicode value of the character at the given index.
concat()	It allows the users to combine two or more strings.
indexOf()	It returns the position of a char value in the specified string.
lastIndexOf()	It locates a char value in the given string by looking up a character from the last position.

String Methods

Methods	Description
search()	It returns the position of the regular expression, if a match is found.
match()	It searches a given string for a specified regular expression and returns that regular expression if a match is found.
replace()	It replaces a string with the replacement specified.
substr()	It is used to retrieve a portion of a string based on the starting position and length specified.
substring()	It is used to retrieve a portion of the given string based on the specified index.

String Methods

Methods	Description
slice()	It is used to get a specific part of a string. It enables us to assign both positive and negative indexes.
toLowerCase()	It converts the given string into lowercase letters.
toLocaleLowerCase()	It converts the given string into lowercase letters using the current locale of the host.
toUpperCase()	It converts the given string into uppercase letters.
toLocaleUpperCase()	It converts the given string to lowercase letters using the current locale of the host.

String Methods

Methods	Description
toString()	It provides a string representing the particular object.
valueOf()	It returns a string that represents the specified object.
split()	It divides a string into substring arrays and returns the resulting array.
trim()	It removes the white space from the string's left and right sides.

Arithmetic Operation

The arithmetic operator is used to perform mathematical operations like addition, subtraction, multiplication, division, and modulus on the given operands.

The addition operator (+) returns the sum of two values. Also, users can use the addition operator with two variables.

The subtraction operator (-) subtracts one number from another.

Arithmetic Operation

The asterisk (*) represents the multiplication operator. It also multiplies two numbers to provide a single result.

The slash (/) character represents the divide operator. The divide operator divides the first value by the second value.

The percent (%) character represents the modulus operator, which results in a remainder.

Arithmetic Operation

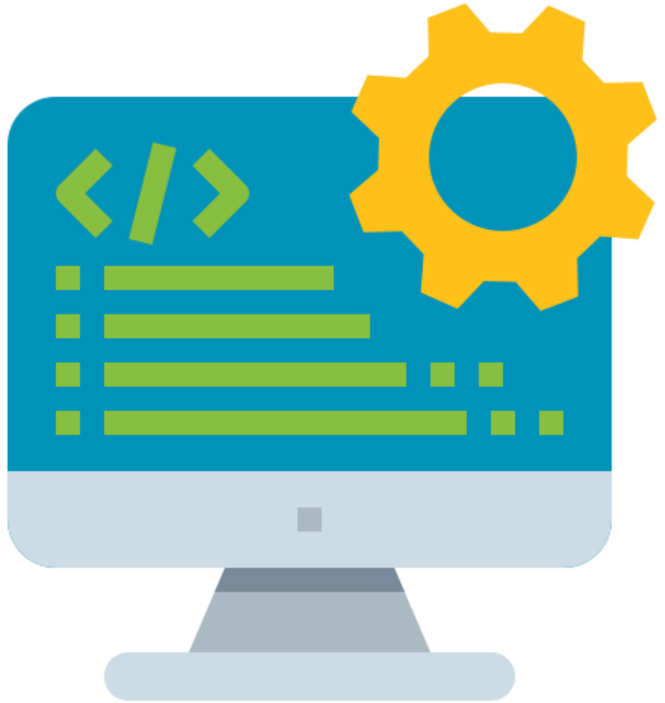
The double asterisk (**) represents the exponentiation operator.

The double plus symbol (++) represents the increment operator.

The double minus symbol (--) represents the decrement operator.

Control Statements

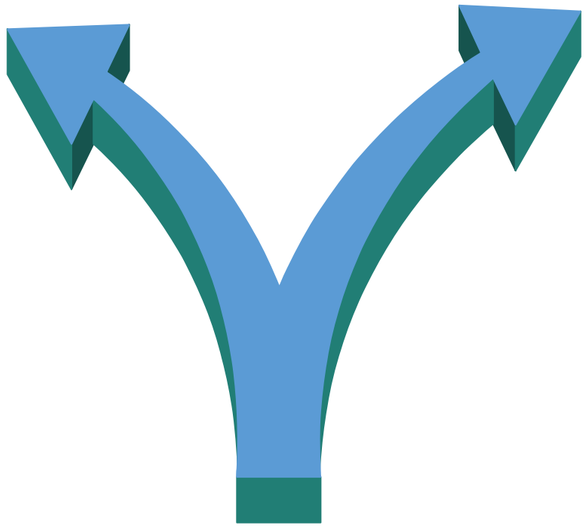
A control statement allows structured control of the sequence of application statements, such as loops and conditional tests.



Control Statements

Control statements are divided into two categories:

Conditional statements

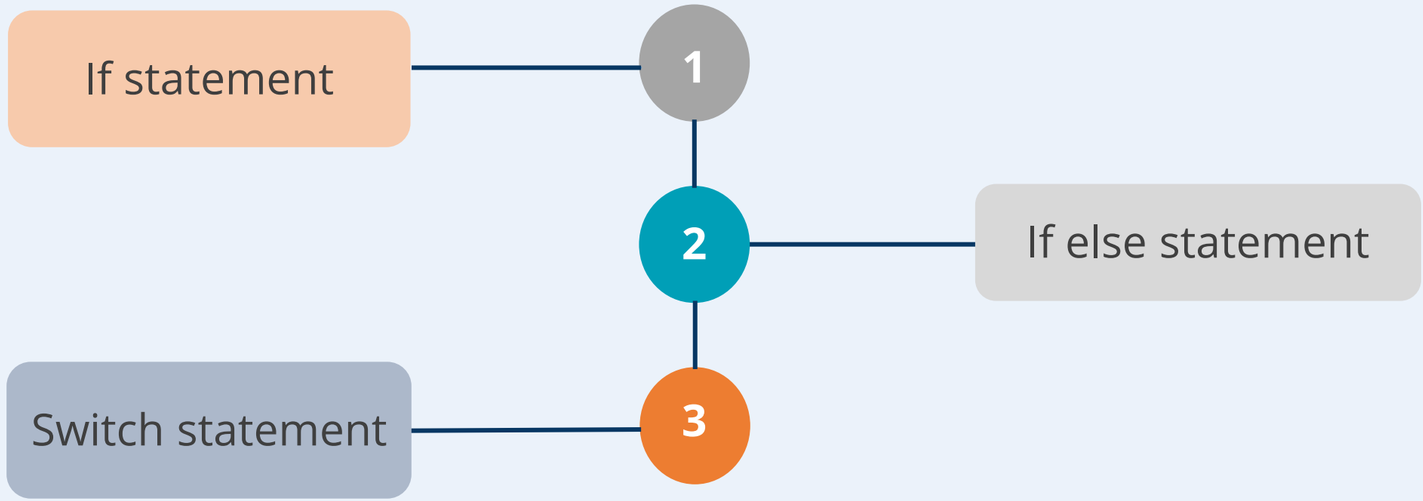


Iterative statements

Conditional Statements

Conditional statements decide the next step based on the result. A conditional statement results in either true or false.

The following are the different types of conditional statements:



Conditional Statements

If statement

It evaluates the content only if the expression is true.

Example:

```
<script>
var a=20;
if(a>10){
document.write("value of a is greater
 than 10");
}
</script>
```

Conditional Statements

If else statement

It evaluates the content whether condition is true of false. The syntax of JavaScript if-else statement is given below.

Example:

```
<script>
var a=20;
if(a%2==0){
document.write("a is even number");
}
else{
document.write("a is odd number");
}
</script>
```

Using If else In Applying Promo Codes

Problem Statement:

You are asked to use If else statement in applying promo codes.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to use If else in applying promo codes are:

1. Use If else

Conditional Statements

Switch statement

The JavaScript switch statement is used to execute one code from multiple expressions. It is just like else if statement.

Example:

```
<script>
var grade='B';
var result;
switch(grade){
case 'A':
result="A Grade";
break;
case 'B’:
result="B Grade";
break;
default:
result="No Grade";
}
document.write(result);
</script>
```

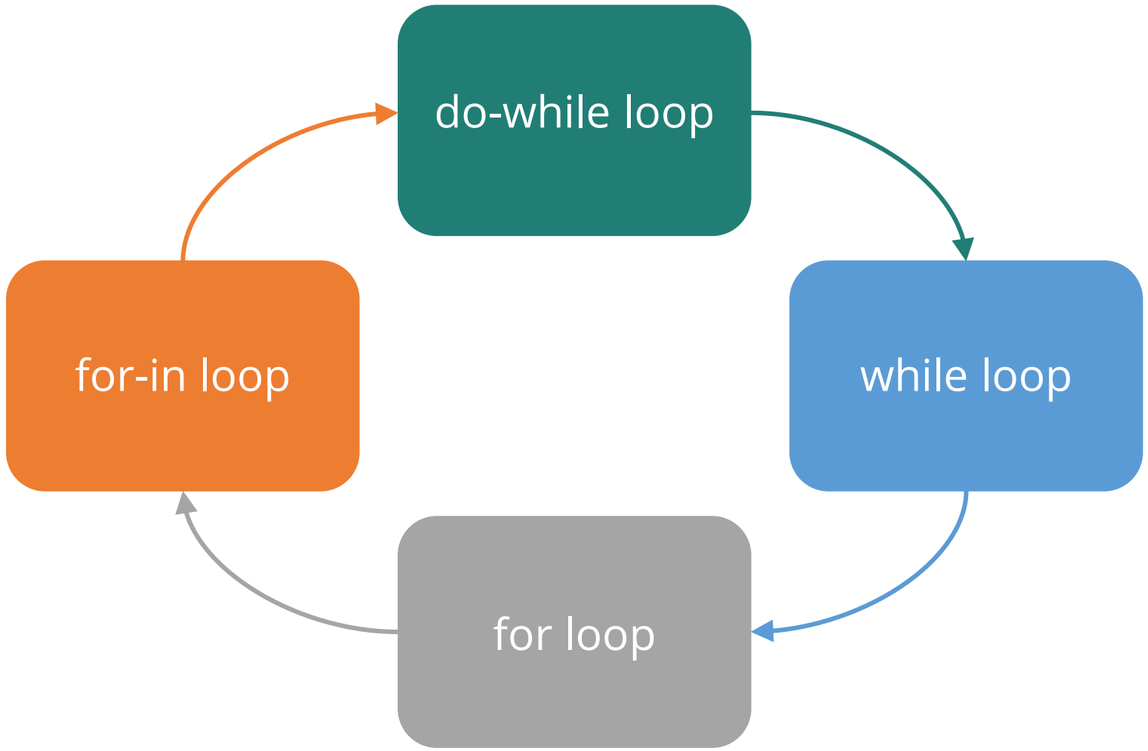
Iterative Statements

Iterative statements (or compound statements) get executed zero or more times and are subject to some loop-termination criteria.

1. While statement
2. Do-While statement
3. For statement

Loop

A loop is a set of instructions that is repeatedly executed until a specific condition is met.



While Loop

The elements are iterated an infinite number of times in the while loop. If the number of iterations is unknown, it should be used.

Example:

```
<script>
var i=11;
while (i<=15)
{
document.write(i + "<br/>");
i++;
}
</script>
```

Do-While Loop

The elements are iterated an infinite number of times in the do-while loop. The code is run at least once regardless of whether the condition is true or false.

Example:

```
<script>
var i=21;
do{
document.write(i + "<br/>");
i++;
}
while (i<=25);
</script>
```

For Loop

The elements are iterated for a set number of times in the for loop. If the number of iterations is known, it should be used.

Example:

```
<script>
for (i=1; i<=5; i++)
{
document.write(i + "<br/>")
}
</script>
```

For-In Loop

The for-in loop is a simple control statement that lets users loop through an object's properties.

Example:

```
<script>
for (let x in location)
{
  text += x + "";
}
</script>
```

Comparator

Comparison operators are used in logical statements to determine equality or a difference between variables or values.

Operator	Description
==	Equal to
===	Equal value and equal type
!=	Not equal
!==	Not equal value or not equal type

Comparator

Comparison operators can be used in conditional statements to compare values and take action depending on the result.

Operator	Description
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Function

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

Syntax:

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

A JavaScript function is defined with the function keyword, followed by a name and parentheses ().

Function Invocation

The code inside the function will execute when something invokes (calls) the function:

01

Automatically (self-invoked)

02

When function is invoked (called) from JavaScript code

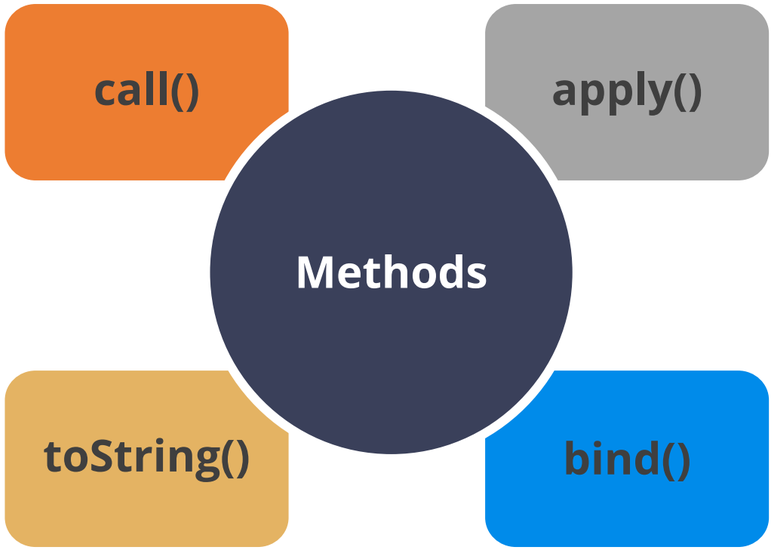
03

When an event occurs (when a user clicks a button)

Function Methods

It is used to call a function that contains this value and an argument list.

It is used to call a function that contains this value and a single array of arguments.



It returns the result in a form of a string.

It is used to create a new function.

Function Objects

The purpose of Function constructor is to create a new function object. It executes the code globally.

Example:

```
<script>
var add=new Function("num1","num2
","return num1+num2");
document.writeln(add(2,5));
</script>
```


However, if a user calls the constructor directly, a function is created dynamically but in an unsecured way.

Array

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together.

1. By array literal
2. By creating an instance of array directly (using new keyword)
3. By using an array constructor (using new keyword)

Array

The simple example of creating an array using an array literal in JavaScript:

Example:

```
<script>
var emp=["Sonoo","Vimal","Ratan"];
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br/>");
}
</script>
```

The values are contained inside [ ] and separated by a comma.

Array

The following is a simple example of creating an array by directly creating an array instance in JavaScript:

Example:

```
<script>
var i;
var emp = new Array();
emp[0] = "Anand";
emp[1] = "Varsha";
emp[2] = "John";
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

A new keyword is used to create an instance of an array.

Array

The following is a simple example of creating an array in JavaScript using an array constructor:

Example:

```
<script>
var emp=new Array("Jai","Vijay","Smit
h");
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

An array object can be generated by passing arguments to the constructor so that the user does not have to explicitly declare values.

Working on Creating Dish and User Object for Food Delivery Application

Problem Statement:

You are required to create dish and user object for food delivery application.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to create dish and user object for food delivery application are:

1. Create dish and user object for food delivery application

Creating a Cart List of Dish Objects

Problem Statement:

You are asked to create a cart list of dish objects.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to create a cart list of dish objects are:

1. Create a cart list of dish objects

Creating Function to Sort the Dish List Based on Price

Problem Statement:

You are asked to create function to sort the dish list based on price.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to create function to sort the dish list based on price are:

1. Create function to sort the dish list based on price

Key Takeaways

- JavaScript is a cross-platform, interpreted, object-oriented scripting language.
- A JavaScript variable is a container, not a value. This means that variables are containers for values.
- Comparison operators return a boolean value. They are widely used with conditional JavaScript statements.
- JavaScript arrays are also objects. It can be verified using the type of operator.