Coding Bootcamp

simplilearn

JavaScript

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    cout << "Hello, world!" << endl;
    return 0;
}
```

Objects

Learning Objectives

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

- Identify the ways to create objects for performing various operations
- Define the factory function syntax for writing reusable code
- Apply factory functions in code for creating multiple similar objects
- Identify the properties of an object for effective data handling
- Implement object cloning techniques for memory management

Learning Objectives

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

- Evaluate the dynamic nature of objects for optimizing code flexibility
- Examine the concepts of garbage collection in JavaScript for managing memory
- Utilize Math and String objects for performing complex calculations and string manipulations
- List JavaScript string methods for efficient string operations
- Employ template literals for embedding expressions and creating multi-line strings

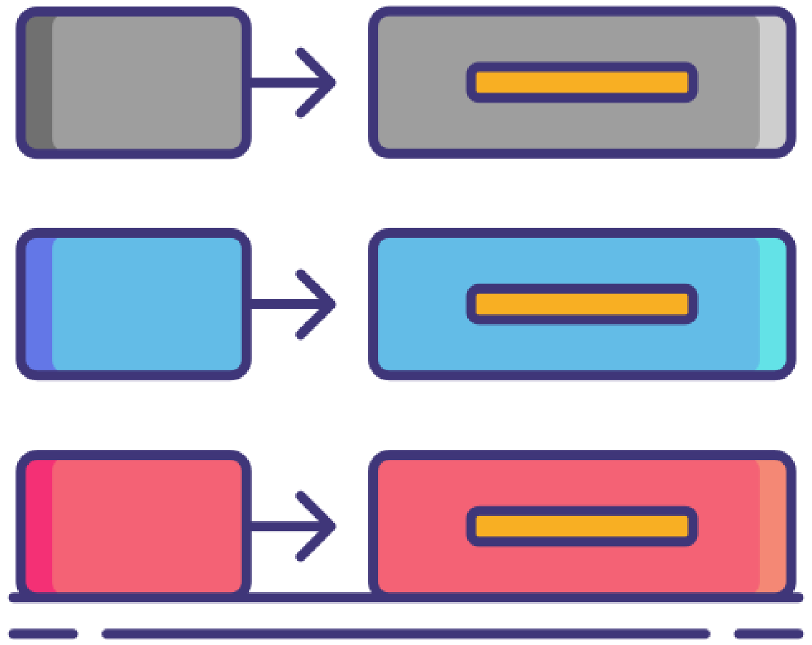
TECHNOLOGY

What Are Objects?

Object

In JavaScript, the object is an unordered standalone entity having property and type.

The object stores the data as a key-value pair.

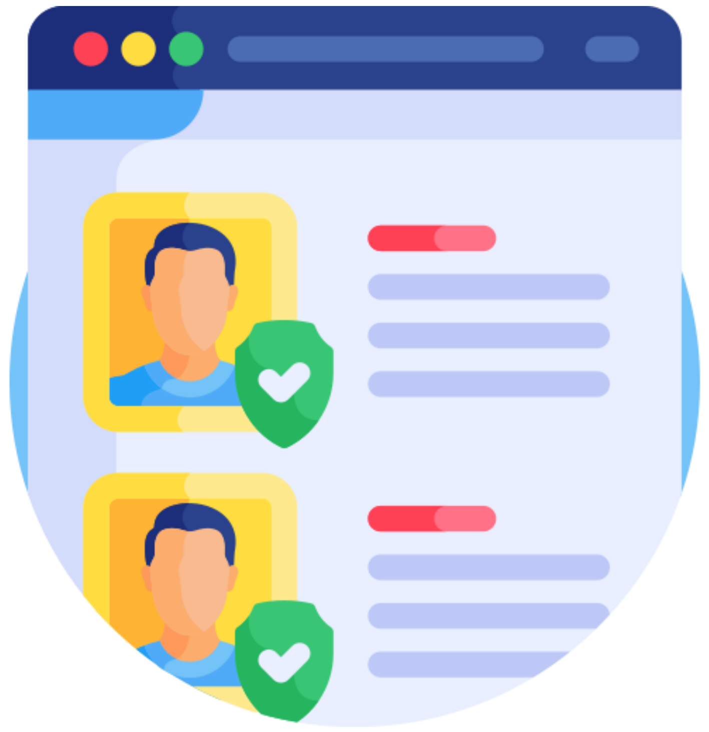


Syntax:

```
let objectName = {
key: value (key and value separated by a colon)
};
```

Object

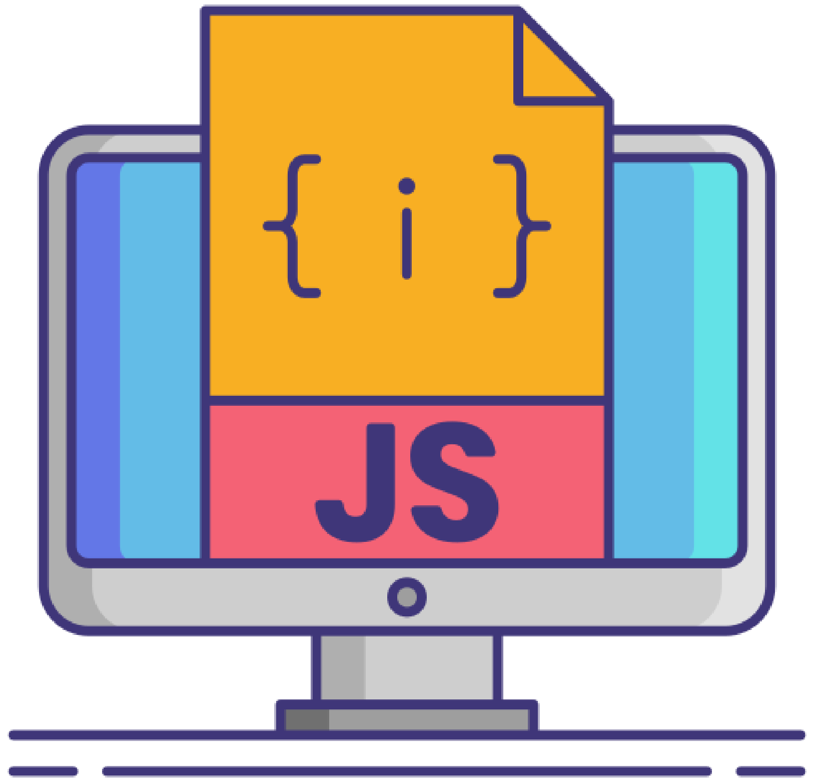
Example:



```
const user = {name:"Mike Hary", email:"mike.hary@example.com", age:30, weight:60};
```

Enumerating Properties

Enumerating properties in JavaScript refers to the process of iterating over the properties of an object to access, list, or manipulate them.

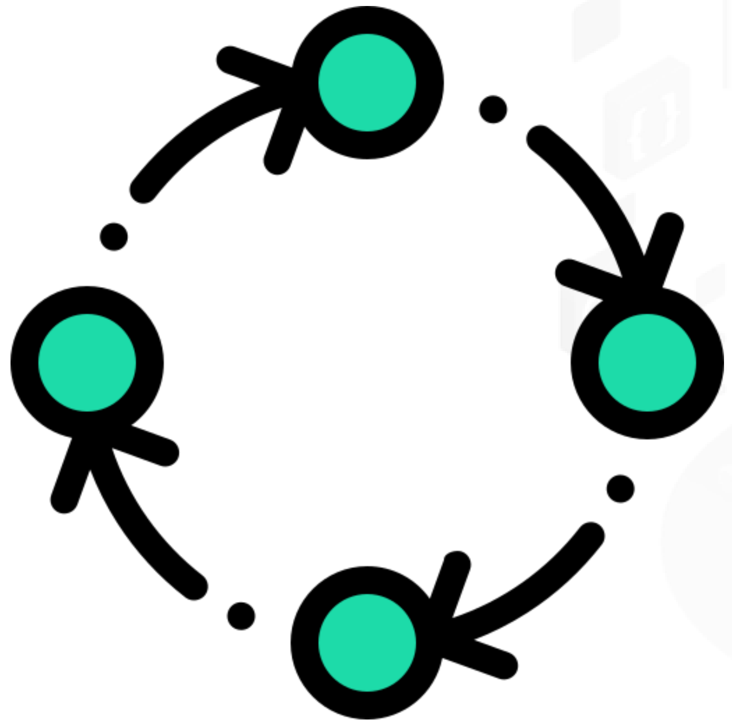


With some approaches or procedures, one may enumerate the characteristics of any given object.

For-in Loop

For-in loop is used to enumerate all the properties and methods in an object.

```
const circle = {
  radius: 2,
  drawpicture() {
    console.log('drawpicture');
  }
};

for (let key in circle) {
  console.log(key, circle[key]);
}
```



Object.keys

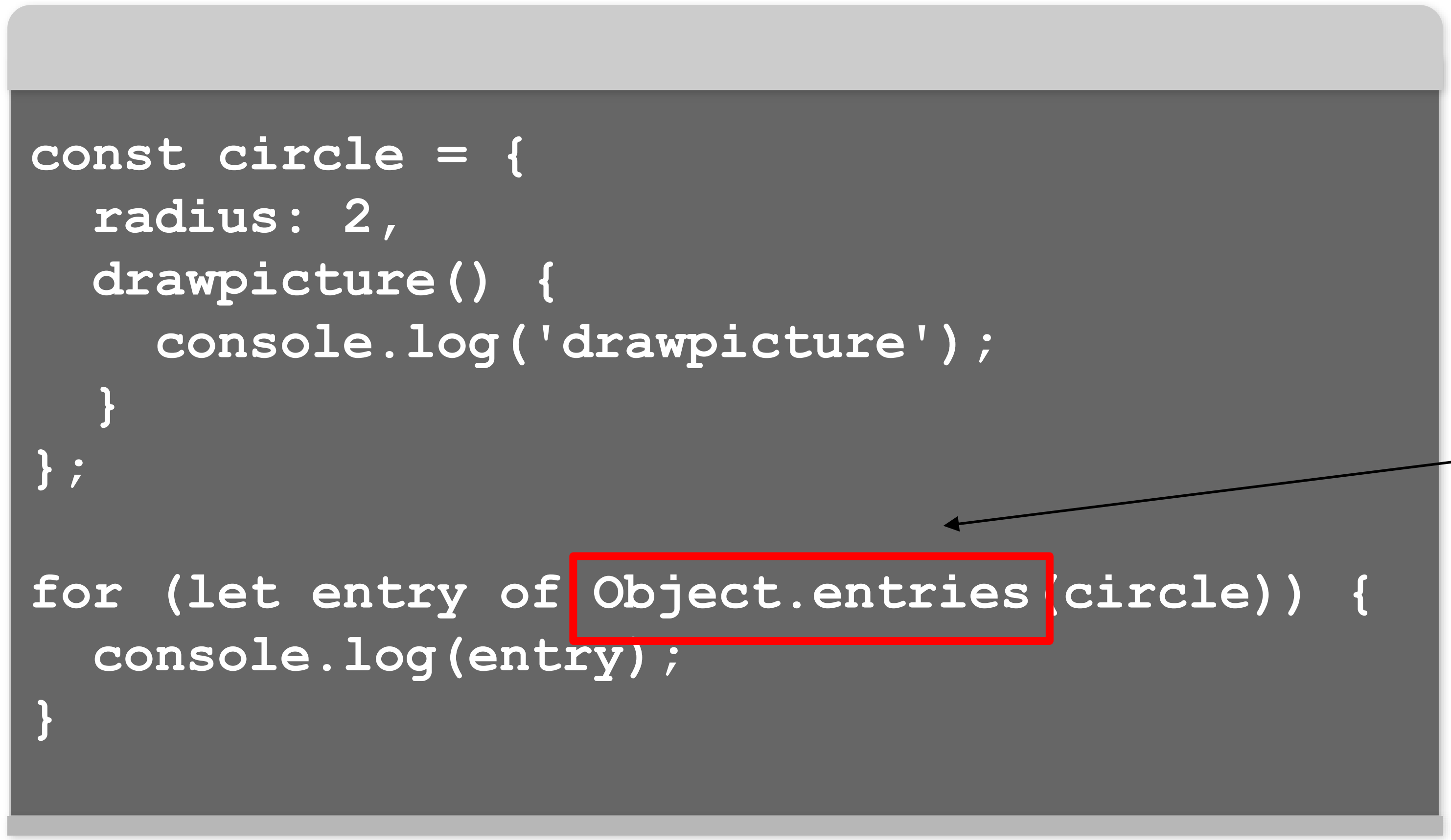
It is a method that returns an array of a given object's enumerable property names (keys).

```
const circle = {
  radius: 2,
  drawpicture() {
    console.log('drawpicture');
  }
};

for (let key of Object.keys(circle)) {
  console.log(key);
}
```

The loop will return the key and the method in the circle object in the string array.

Object.entries

It is a method that returns an array of the given object's enumerable property [key, value] pairs.



```
const circle = {
  radius: 2,
  drawpicture() {
    console.log('drawpicture');
  }
};

for (let entry of Object.entries(circle)) {
  console.log(entry);
}
```

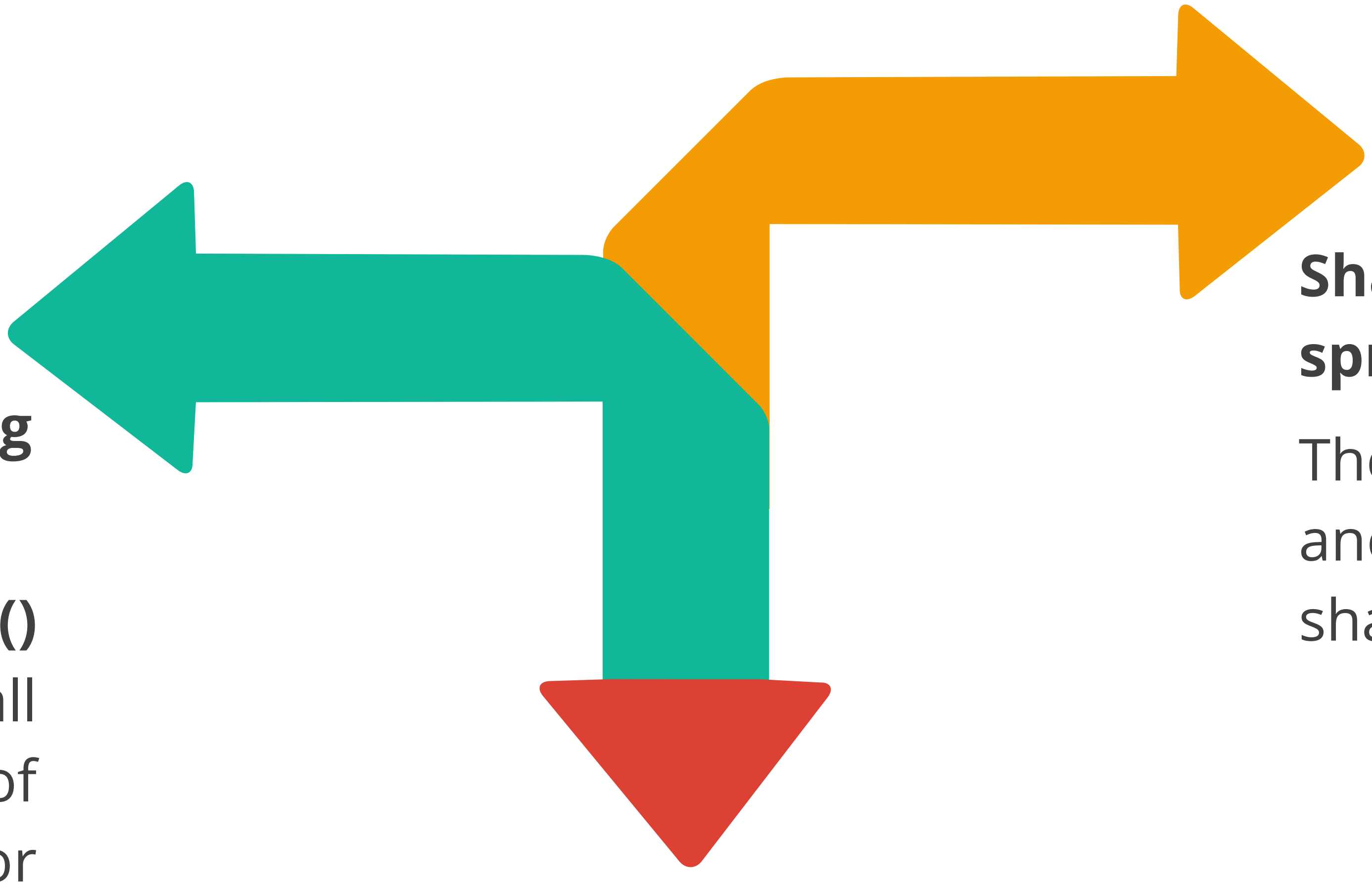
It shows the properties of a given object.

Cloning an Object

Cloning an object in JavaScript involves creating a new object with the same properties and values as an existing object.

This is done to avoid directly modifying the original object, thus preventing unintended side effects or bugs due to shared references.

Methods of Cloning an Object



Shallow clone using Object.assign():

The **Object.assign()** method copies all enumerable properties of an object from one or more source objects to a target object.

Shallow clone using spread operator(...):

The **spread operator (...)** is another way to perform a shallow copy of an object.

Deep clone using JSON Methods:

For a deep clone, the **JSON.stringify()** and **JSON.parse()** methods can be used.

Creating Product and User Objects for eCommerce Store

Problem Statement:

You have been asked to demonstrate the creation of objects in JavaScript and their usage in an eCommerce store scenario.

Outcome:

By completing this task, you demonstrated creating JavaScript objects for an eCommerce store: a user object for customer information and a product object for item details.

Note: Refer to the demo document for detailed steps:
01_Creating_Product_and_User_Objects_for_eCommerce_Store

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Create a user object
2. Create a product object

Factory and Constructor Function

Factory Function

It is a type of function that creates and returns a new object.

Function

```
let rose = {
  firstName: 'Roselin',
  lastName: 'Joseph',
  getFullName() {
    return this.firstName + ' ' +
this.lastName;
  }
};

console.log(rose.getFullName());
// Output: Roselin Joseph
```

Factory function

```
function createPerson(firstName, lastName) {
  return {
    firstName: firstName,
    lastName: lastName,
    getFullName() {
      return this.firstName + ' ' +
this.lastName;
    }
  };
}

const rose = createPerson('Roselin',
'Joseph');
console.log(rose.getFullName()); // Output:
Roselin Joseph
```

Constructor Function

It is used to create and initialize objects.

```
function Student() {
  this.name = 'Reet';
  this.age = 25;
}

// Create an object
const student = new Student();
console.log(student.name); // Output: Reet
console.log(student.age);  // Output: 25
```

Constructor functions use the ‘new’ keyword to create an object.

Constructor Function

The constructor function is used to create multiple similar objects.

Example:

```
function Vehicle(make, model) {
  this.make = make;
  this.model = model;
}

// Creating multiple similar objects
let vehicle1 = new Vehicle('Honda', 'Activa');
let vehicle2 = new Vehicle('Bajaj', 'Super');

console.log(vehicle1.make);  // Output: Honda
console.log(vehicle1.model); // Output: Activa
console.log(vehicle2.make);  // Output: Bajaj
console.log(vehicle2.model); // Output: Super
```

Constructor Function

The function **Object() { [native code] }** property returns a reference to the instance object.

✓ The constructor value is a reference to the function itself.

✓ All objects by default have a constructor property.

Constructor Property

The constructor property in JavaScript returns a reference to the function that created the instance of an object. This property is inherent to all objects and can be used to identify the type of the object.

Syntax

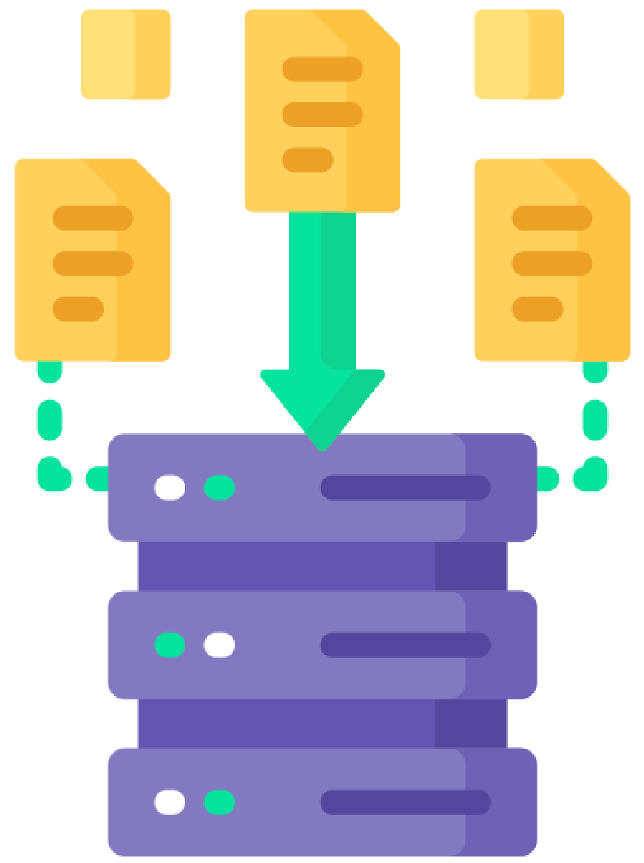
```
let o = {}
o.constructor === Object // true

let o = new Object
o.constructor === Object // true

let a = []
a.constructor === Array // true
```

Value and Reference Types

Value Types

They store the actual data within their memory allocation. A copy of the value is made when a value type variable is assigned to another variable. Examples of value types are:

- Numeric data types
- Boolean (true or false)
- Null and undefined
- String and symbol



Value Types: Example

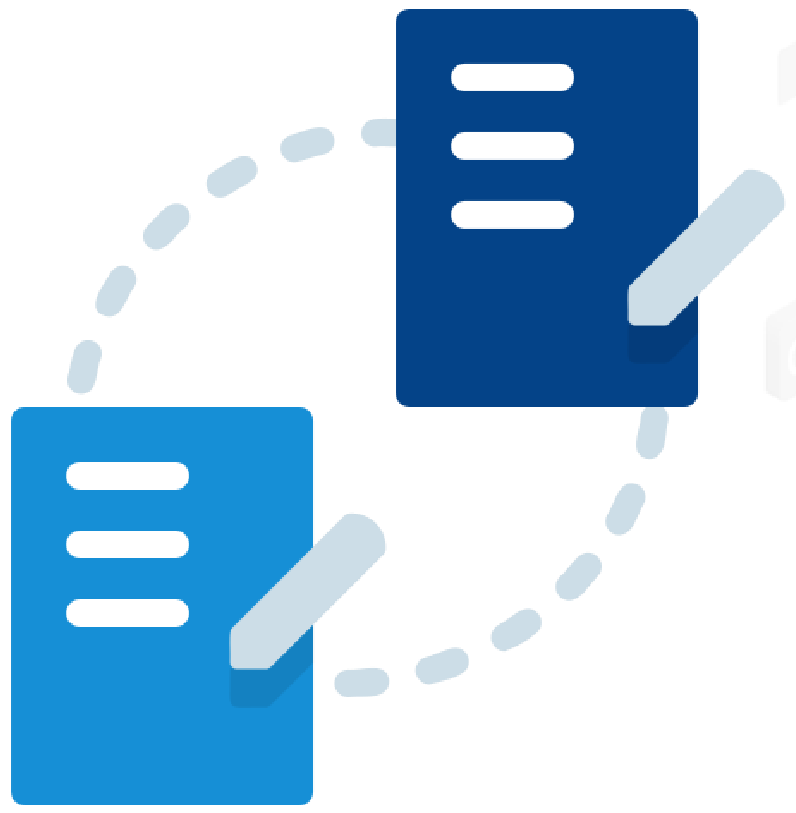
```
let a = 10;
let b = a; // b gets a copy of a's value

a = 20;

console.log(a); // Output: 20
console.log(b); // Output: 10
```

In the above example, **b** gets a copy of **a**'s value.
Changing **a** afterward does not affect **b**.

Reference Types

They store a reference or address to the actual data. When a reference type variable is assigned to another variable, both variables point to the same object in memory. Examples of reference types are:

1. Objects ({})
2. Arrays ([])
3. Functions



Reference Types: Example

```
let obj1 = { name: 'Alice' };
let obj2 = obj1; // obj2 points to the same object as obj1

obj1.name = 'Bob';

console.log(obj1.name); // Output: Bob
console.log(obj2.name); // Output: Bob
```

In the example above, **obj1** and **obj2** both refer to the same object.
Thus, changing a property of the object through **obj1** will also change it for **obj2**.

TECHNOLOGY

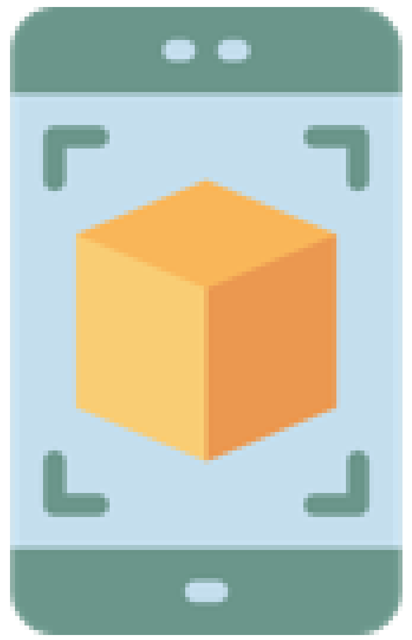
Garbage Collection

What is Garbage Collection?

It refers to the process of identifying and reclaiming memory that is no longer accessible or needed by the application.

✓ It keeps track of memory allocation.

✓ It makes decisions around recovering unused memory.

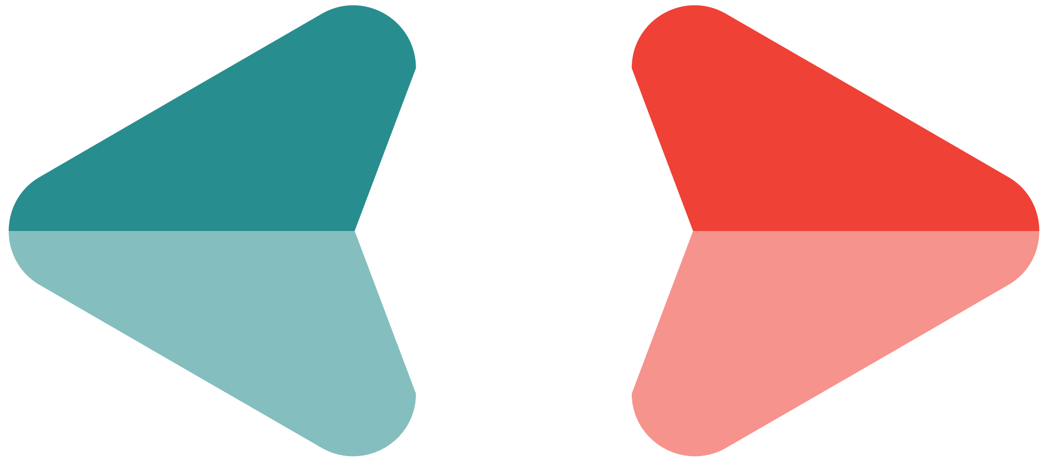


Garbage Collection

Garbage collectors in JavaScript use algorithms to track memory allocation and determine when an object is no longer needed. The most common algorithm is the **mark-and-sweep** algorithm:

Mark phase

The garbage collector traverses the object graph starting from the root (global object, local variables, etc.) and marks all reachable objects.



Sweep phase

The garbage collector then scans the memory and reclaims the memory occupied by objects that are not marked as reachable, effectively freeing up space for future allocations.

Garbage Collection: Example

Consider the following example to illustrate garbage collection:

```
function createObject() {
  let obj = { name: "Temporary Object" };
  console.log(obj.name); // Output: Temporary Object
}

createObject();
```

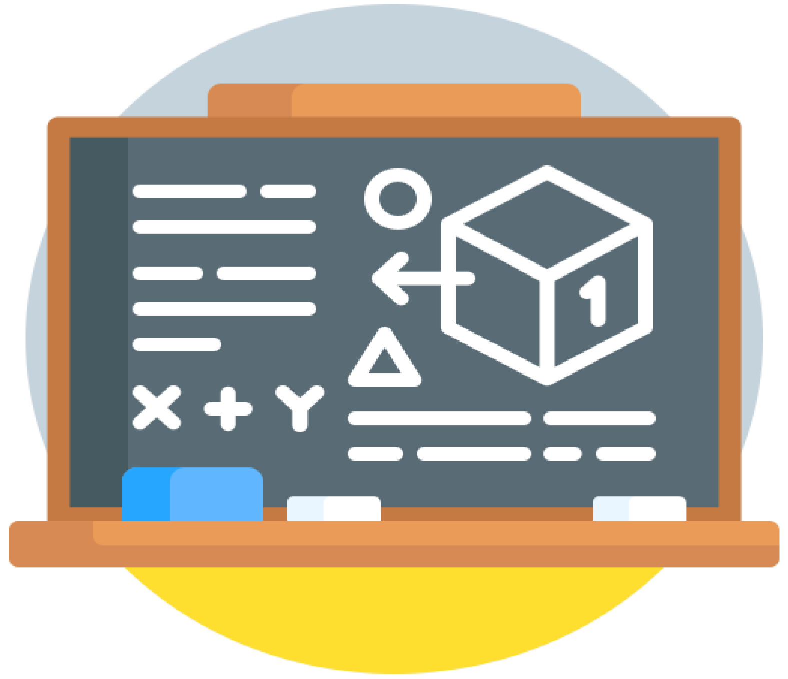
In the above example, after the function execution, **obj** is no longer accessible. The garbage collector will reclaim the memory allocated to **obj.**

TECHNOLOGY

Math Object

What Is Math Object?

It is a built-in object that provides properties and methods for performing mathematical operations.



It includes a wide range of mathematical functions, constants, and utilities that can be used for complex calculations.

Math Object: Methods

The types of math object methods are:

Method	Description
abs()	Returns the absolute value of the given number
acos()	Returns the arccosine of the given number in radians
asin()	Returns the arcsine of the given number in radians
atan()	Returns the arctangent of the given number in radians
cbrt()	Returns the cube root of the given number
ceil()	Returns the smallest integer value greater than or equal to the given number
cos()	Returns the cosine of the given number
cosh()	Returns the hyperbolic cosine of the given number

Math Object: Methods

The types of math object methods are:

Method	Description
exp()	Returns the exponential form of the given number
floor()	Returns the largest integer value lower than or equal to the given number
hypot()	Returns the square root of the sum of the squares of the given numbers
log()	Returns the natural logarithm of a number
max()	Returns the maximum value of the given numbers
min()	Returns the minimum value of the given numbers
pow()	Returns the value of the base to the power of the exponent
random()	Returns the random number between 0 and 1

Math Object: Methods

The types of math object methods are:

Method	Description
round()	Returns the closest integer value of the given number
sign()	Returns the sign of the given number
sin()	Returns the sine of the given number
sinh()	Returns the hyperbolic sine of the given number
sqrt()	Returns the square root of the given number
tan()	Returns the tangent of the given number
tanh()	Returns the hyperbolic tangent of the given number
trunc()	Returns an integer part of the given number

Math Object: Example

```
<!DOCTYPE html>
<html>
<body>
  Ceil of 4.2 is: <span id="result"></span>
  <script>
    document.getElementById('result').innerHTML =
Math.ceil(4.2);
  </script>
</body>
</html>
```

Output:

```
Ceil of 4.2 is: 5
```

String

What Is a String?

In JavaScript, a string is a sequence of characters used to represent and manipulate text. They are one of the fundamental data types in JavaScript and can be created in two primary ways:

- ✓ By string literal
- ✓ By string object

String Literals

A **string literal** is the simplest and most common way to create a string. They are enclosed in single quotes ('...'), double quotes ("..."), or backticks (`...`).

```
let singleQuoteString = 'Hello, World!';
let doubleQuoteString = "Hello, World!";
let templateLiteralString = `Hello, World!`;
```

Example

```
let name = "Alice";
let greeting = `Hello, ${name}!`; // Embedded
expression
console.log(greeting);
```

Output

```
Hello, Alice!
```

String Object

A **string object** is created using the **String** constructor. This method wraps the string literal in an object, providing additional properties and methods.

```
let stringObject = new String("Hello, World!");
```

Example

```
<!DOCTYPE html>
<html>
<body>
  <script>
    var stringName = new String("Hello World!");
    document.write(stringName);
  </script>
</body>
</html>
```

Output

```
Hello World!
```

TECHNOLOGY

JavaScript String Methods

JavaScript String Methods

Method	Description
charAt()	It provides the character value present at the specified index.
charCode()	It provides the unicode value of a character present at the specified index.
concat()	It provides the combination of two or more strings.
indexOf()	It provides the position of a char value present in the given string.
lastIndex()	It provides the position of a char value present in the given string by searching for a character from the last position.
search()	It searches a specified regular expression in a given string and returns its position if the match occurs in it.

JavaScript String Methods

Method	Description
match()	It searches for a specified regular expression in each string and returns the regular expression if a match occurs.
replace()	It replaces a given string with the specified replacement.
substr()	It is used to fetch the part of the given string based on the specified starting position and the length.
substring()	It is used to fetch the part of the given string based on the specified index.
slice()	It is used to fetch a part of the given string and allows you to assign a positive and negative index.
toLowerCase()	It changes the uppercase letter in the given string to a lowercase letter.

JavaScript String Methods

Method	Description
toLocaleLowerCase()	It changes the uppercase letter in the given string to a lowercase letter based on the host's current locale.
toUpperCase()	It converts the given string into an uppercase letter.
toLocaleUpperCase()	It converts the given string into an uppercase letter based on the host’s current locale.
toString()	It provides a string representing a particular object.
valueOf()	It provides the primitive value of the string object.
split()	It splits a string into a substring array and then returns the newly created array.
trim()	It trims the white space from the left and right-side strings.

JavaScript String Methods: Example

The provided example demonstrates how to use the charAt() method of the JavaScript String object to retrieve a character at a specific index in a string.

Example

```
<!DOCTYPE html>
<html>
<body>
<script>
var str = "Javascript string methods"
document.write(str.charAt(8));
</script>
</body>
</html>
```

Output

```
i
```

Using Various String Built-in Methods

Problem Statement:

You have been asked to demonstrate the usage of various built-in string methods in JavaScript for string manipulation and analysis

Outcome:

By completing this task, you will be able to learn the usage of various built-in string methods in JavaScript for string manipulation and analysis.

Note: Refer to the demo document for detailed steps:
02_Using_Various_String_Built-in_Methods

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Explore built-in string methods

TECHNOLOGY

Template Literals

What Are Template Literals?

They are a feature in JavaScript that allows for easier and more readable string creation and manipulation. They are defined using backticks (\` \`) instead of single or double quotes.

Example

```
let newString = `Template Literals string`;
console.log(newString);
```

Output

```
Template Literals string
```

Differentiating Between Value and Reference When Creating Storage Containers

Problem Statement:

You have been asked to understand the difference between value and reference while creating storage containers in JavaScript.

Outcome:

By completing this task, you will be able to gain an understanding of the difference between value and reference when creating storage containers in JavaScript.

Note: Refer to the demo document for detailed steps:
03_Differentiating_Between_Value_and_Reference_When_Creating_Storage_Containers

Assisted Practice: Guidelines

Steps to be followed are:

1. Analyze the concept of value versus reference

Key Takeaways

- In JavaScript, an object is an unordered, standalone entity with property and type.
- Object.keys is a method that returns an array of a given object's enumerable property names (keys).
- The constructor function is used to create and initialize objects.
- Garbage collection is the process of identifying and reclaiming memory that is no longer needed by the application.
- Template literals are a feature in JavaScript that allows for easier and more readable string creation and manipulation.

TECHNOLOGY

Thank You