## Nested Objects:

Write a Program to access properties of given nested object?

const person = {

  name: 'John',

  age: 20,

  marks: {

      science: 70,

      math: 75

+

const person = {

  name: 'John',

  age: 20,

  marks: {

      science: 70,

      math: 75

  }

}

// accessing property of student object

console.log(student.marks); // {science: 70, math: 75}

// accessing property of marks object

console.log(student.marks.science); // 70

## Using object literal

// program to create JavaScript object using object literal

const person = {

  name: 'Joh',

  age: 20,

  hobbies: ['reading', 'games', 'coding'],

  greet: function() {

      console.log('Hello everyone.');

  },

  score: {

      maths: 90,

      science: 80

  }

};

console.log(typeof person); // object

// accessing the object value

console.log(person.name);

console.log(person.hobbies[0]);

person.greet();

console.log(person.score.maths);

## Example 2: Create an Object using Instance of Object Directly

// program to create JavaScript object using instance of an object

const person = new Object ( {

  name: 'John',

  age: 20,

  hobbies: ['reading', 'games', 'coding'],

  greet: function() {

      console.log('Hello everyone.');

  },

  score: {

      maths: 90,

      science: 80

  }

});

console.log(typeof person); // object

// accessing the object value

console.log(person.name);

console.log(person.hobbies[0]);

person.greet();

console.log(person.score.maths);

**Create an object using Constructor Function**

// program to create JavaScript object using instance of an object

function Person() {

  this.name = 'John',

  this.age = 20,

  this.hobbies = ['reading', 'games', 'coding'],

  this.greet = function() {

      console.log('Hello everyone.');

  },

  this.score = {

      maths: 90,

      science: 80

  }

}

const person = new Person();

console.log(typeof person); // object

// accessing the object value

console.log(person.name);

console.log(person.hobbies[0]);

person.greet();

console.log(person.score.maths);

# JavaScript Object.assign()

const obj1 = { a: 1 };

const obj2 = { b: 2 };

const obj3 = { c: 3 };

// combine all the properties of

// obj1, obj2, obj3 into a single object

const mergedObj = Object.assign(obj1, obj2, obj3);

console.log(mergedObj);

// Output: { a: 1, b: 2, c: 3 }

# Javascript Object.create()

let Student = {

  name: "Lisa",

  age: 24,

  marks: 78.9,

  display() {

    console.log("Name:", this.name);

  }

};

// create object from Student prototype

let std1 = Object.create(Student);

std1.name = "Sheeran";

std1.display();

// Output: Name: Sheeran

// Array objects

const arr = ["A", "B", "C"];

console.log(Object.keys(arr)); // ['0', '1', '2']

// array-like objects

const obj = { 65: "A", 66: "B", 67: "C" };

console.log(Object.keys(obj)); // ['65', '66', '67']

// random key ordering

const obj1 = { 42: "a", 22: "b", 71: "c" };

console.log(Object.keys(obj1)); // ['22', '42', '71']

// string => from ES2015+, non objects are coerced to object

const string = "code";

console.log(Object.keys(string)); // [ '0', '1', '2', '3' ]