❖ <u>**JavaScript is a functional programing?**</u>

JavaScript is a multi-paradigm language. It can't be cleanly categorized as either an object-oriented or a functional programming language. That's because one can easily use whichever paradigms one wants in JavaScript.

JavaScript can be shown to have elements of both OOP and functional programming. You can use objects and prototypes in it and do all manner of OOP-based computation. And, you can also use first-class functions and immutable objects in it.

JavaScript, like many other languages, lends itself to multiple programming paradigms. It's up to the developer to decide which paradigm is better suited to the problem at hand.

❖ <u>**Differences between shallow copy and deep copy:**</u>

A shallow copy is a copy that only goes one level deep. In other words, it copies the object and all its properties, but any nested objects or arrays will still reference the same memory location as the original object. It means that if you make changes to the nested object, it will also affect the original object, as well as the copied object.

- ex: var original={a:1,b:{c:3}}
  var shallow_copy={...original}

var original={name:hager,age:19}
var shallow_copy=Object.assign({},original)

The spread operator ... is used to create a new object shallowCopy that is a shallow copy of original.

A deep copy is a copy that creates a new object with new memory locations for all of its properties and nested objects or arrays. It means that if you make changes to the copied object or any of its nested objects or arrays, it will not affect the original object.

- ex: var original={a:1,b:{c:3}}
  var deep_copy=JSON.parse(JSON.stringify(original))

  we use JSON.stringify() to convert original to a JSON string, and then use JSON.parse() to convert that string back to a new object deepCopy

In JavaScript, there are two ways to copy objects: shallow copy and deep copy. Shallow copying creates a new object with references to the same memory locations as the original object, while deep copying creates a new object with new memory locations for all of its properties and nested objects or arrays.

Shallow copying can be more efficient in terms of performance, but may result in unexpected behavior if changes to a copied object affect the original object. Deep copying ensures that changes to a copied object do not affect the original object, but may be more expensive in terms of performance.

While using JSON.parse() and JSON.stringify() is an easy way to create a deep copy of an object, it may not work in all cases. If you need to create a deep copy of an object, using

JSON.parse() and JSON.stringify() is an easy option. However, if the object being copied contains functions or circular references, a recursive deep copy function may be necessary.

- ❖ **<u>Imperative Programming and Declarative Programming:</u>**
  Imperative Programming as the name suggests is a type of programming paradigm that describes how the program executes. Developers are more concerned with how to get an answer step by step. It comprises the sequence of command imperatives. In this, the order of execution is very important and uses both mutable and immutable data. Fortran, Java, C, C++ programming languages are examples of imperative programming.

Declarative Programming as the name suggests is a type of programming paradigm that describes what programs to be executed. Developers are more concerned with the answer that is received. It declares what kind of results we want and leave programming language aside focusing on simply figuring out how to produce them. In simple words, it mainly focuses on end result. It expresses the logic of computation. Miranda, Erlang, Haskell, Prolog are a few popular examples of declarative programming.