Q No 1: Multiple inheritance in JS?

**JavaScript** does **not** support **multiple inheritance**. ... Because an object has a single associated prototype, **JavaScript** cannot dynamically inherit from more than one prototype chain. In **JavaScript**, you can have a constructor function call more than one other constructor function within it.

Q No 2:

Multilevel Inheritance Possible or not in JS?

Ans: - JS does not support multilevel inheritance.

Q No 3:

How can we achieve overloading functionalities in JS?

There are multiple aspects to argument overloading in JavaScript:

1. **Variable arguments** - You can pass different sets of arguments (in both type and quantity) and the function will behave in a way that matches the arguments passed to it.
2. **Default arguments** - You can define a default value for an argument if it is not passed.
3. **Named arguments** - Argument order becomes irrelevant and you just name which arguments you want to pass to the function.

**1: Variable Arguments:**

In implementing these types of overloads, you have several different techniques you can use:

1. You can check for the presence of any given argument by checking to see if the declared argument name value is undefined.
2. You can check the total quantity or arguments with arguments.length.
3. You can check the type of any given argument.
4. For variable numbers of arguments, you can use the arguments pseudo-array to access any given argument with arguments[i].

Here are some examples:

Let's look at jQuery's obj.data() method. It supports four different forms of usage:

obj.data("key");

obj.data("key", value);

obj.data();

obj.data(object);

// get the data element associated with a particular key value

obj.data("key");

If the first argument passed to .data() is a string and the second argument is undefined, then the caller must be using this form.

// set the value associated with a particular key

obj.data("key", value);

If the second argument is not undefined, then set the value of a particular key.

// get all keys/values

obj.data();

If no arguments are passed, then return all keys/values in a returned object.

// set all keys/values from the passed in object

obj.data(object);

**2: Default Arguments:**

In ES6, JavaScript adds built-in language support for default values for arguments.

For example:

function multiply(a, b = 1) {

return a\*b;

}

multiply(5); // 5

**3: Named Arguments:**

Other languages (like Python) allow one to pass named arguments as a means of passing only some arguments and making the arguments independent of the order they are passed in. JavaScript does not directly support the feature of named arguments. A design pattern that is commonly used in its place is to pass a map of properties/values. This can be done by passing an object with properties and values or in ES6 and above, you could actually pass a Map object itself.

Here's a simple ES5 example:

JQuery’s $.ajax () accepts a form of usage where you just pass it a single parameter which is a regular JavaScript object with properties and values. Which properties you pass it determine which arguments/options are being passed to the ajax call. Some may be required, many are optional. Since they are properties on an object, there is no specific order. In fact, there are more than 30 different properties that can be passed on that object, only one (the url) is required.

Here's an example:

$.ajax({url: "http://www.example.com/somepath", data: myArgs, dataType: "json"}).then(function(result) {

// process result here

});