* Call, apply and bind method...

→ Objects can borrow methods and we will see that →

→ It can be done with the help of call, apply and bind methods.

→ These are the methods that are been applied on the method.

→ How it is been possible in the case of Javascript is that we are calling a method on a method is because "functions are considered as objects in javascript."

* → While using 'call' we have to pass the parameters inside the function at the time of calling where we have written the statement.

Ex →
```
var rutvi = {
    name: "Rutvi",
    Profession: "Student",
    workingBio function (experience, age)
{
 Console.log (this.name + " " + this.Profession
        + " " + this.experience + " "
        + this.age);
  }
}

var john = {
    name: "John",
    profession: "Teacher",
}

                call
rutvi.workingBio.(john, 5, 30);
```

[O/P → John Teacher 5 30.]

* Whereas, in the case of 'apply' parameters are been passed in the form of an array.

Ex → rutvi.WorkingBio.apply(john, [6, 30]);

* But in the case of 'bind', it just binds the object.

→ In bind we the flexibility of calling that function whenever we need.

Ex → var bio = rutvi.workingBio(john)

bio(6, 25);

or (If you are sure about parameters)

var bio = rutvi.workingBio(john, 6, 25);

bio();

Note → We can also access a function with call, apply and bind which is defined outside of the object.

* Constructor functions →

→ So, as we were creating different object literals with the properties and methods similar to one another, what happens is, it takes extra efforts and the same piece of code has to be written again and again.

→ So to avoid this what can be done is →

```
function Person(name, profession, hobbies)
{
    this.name = name;
    this.profession = profession;
    this.hobbies = hobbies;
}

var ram = new Person("Ram", "Teacher",
                     ["singing", "Dancing"]);

var sita = new Person("Sita", "Doctor",
                      ["Reading"]);

Console.log(ram);
Console.log(sita);
```

So, in the previous example what happens is →

1. An empty object is created.
2. this keyword is associated to the object that is created.
3. Prototype objects are linked.
4. New created object is returned.

* The function is called using "new" keyword.