Inheritance in Constructor Functions

Inheriting a previously defined constructor function means using the parameters of the previously defined function along with adding some new parameters to the newly defined constructor function. For this, we need to use the call() function which allows us to call a function defined somewhere else in the current context.

Syntax:

```
myFunction.call( this, property1, property2, ... , propertyN )
```

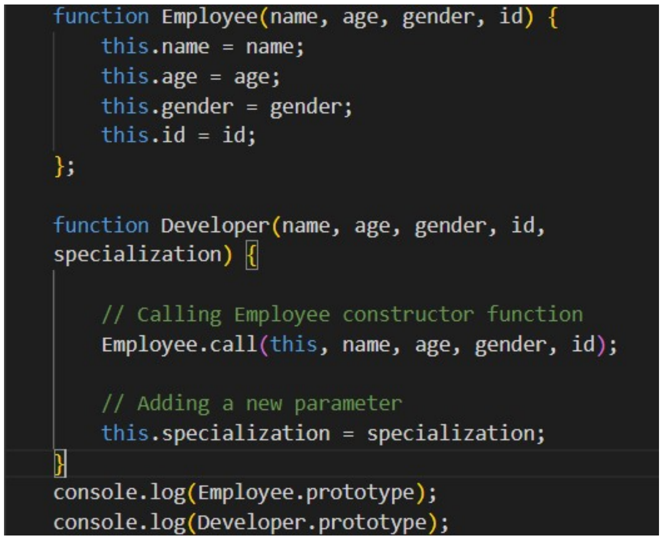
Parameter values:

- **myFunction:** This is a constructor function from which we want to inherit the parameters in a new constructor function.
- **this**: The values of parameters that will use this keyword while calling myFunction.
- **property1, property2, …, propertyN:** The parameters that are to be inherited in the new constructor function.

Return type:

A constructor function or the function which has inherited its properties does not have any return type. It specifies the **prototype** of the properties an object will contain which are created from that constructor function.

Example:

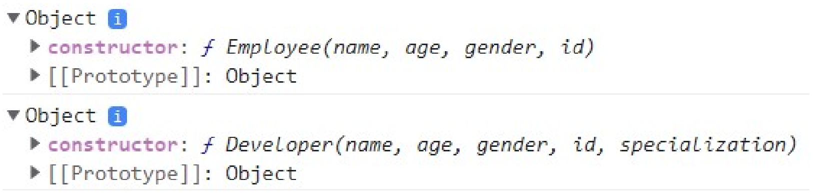


```
function Employee(name, age, gender, id) {
    this.name = name;
    this.age = age;
    this.gender = gender;
    this.id = id;
};

function Developer(name, age, gender, id,
specialization) {

    // Calling Employee constructor function
    Employee.call(this, name, age, gender, id);

    // Adding a new parameter
    this.specialization = specialization;
}
console.log(Employee.prototype);
console.log(Developer.prototype);
```



```
▼ Object
  ▶ constructor: ƒ Employee(name, age, gender, id)
  ▶ [[Prototype]]: Object
▼ Object
  ▶ constructor: ƒ Developer(name, age, gender, id, specialization)
  ▶ [[Prototype]]: Object
```