Advance JavaScript

Object Oriented Programming

Lesson - 02

Objective

- **At the end of this lesson participants will be able to –**
  - Implement Inheritance using JavaScript

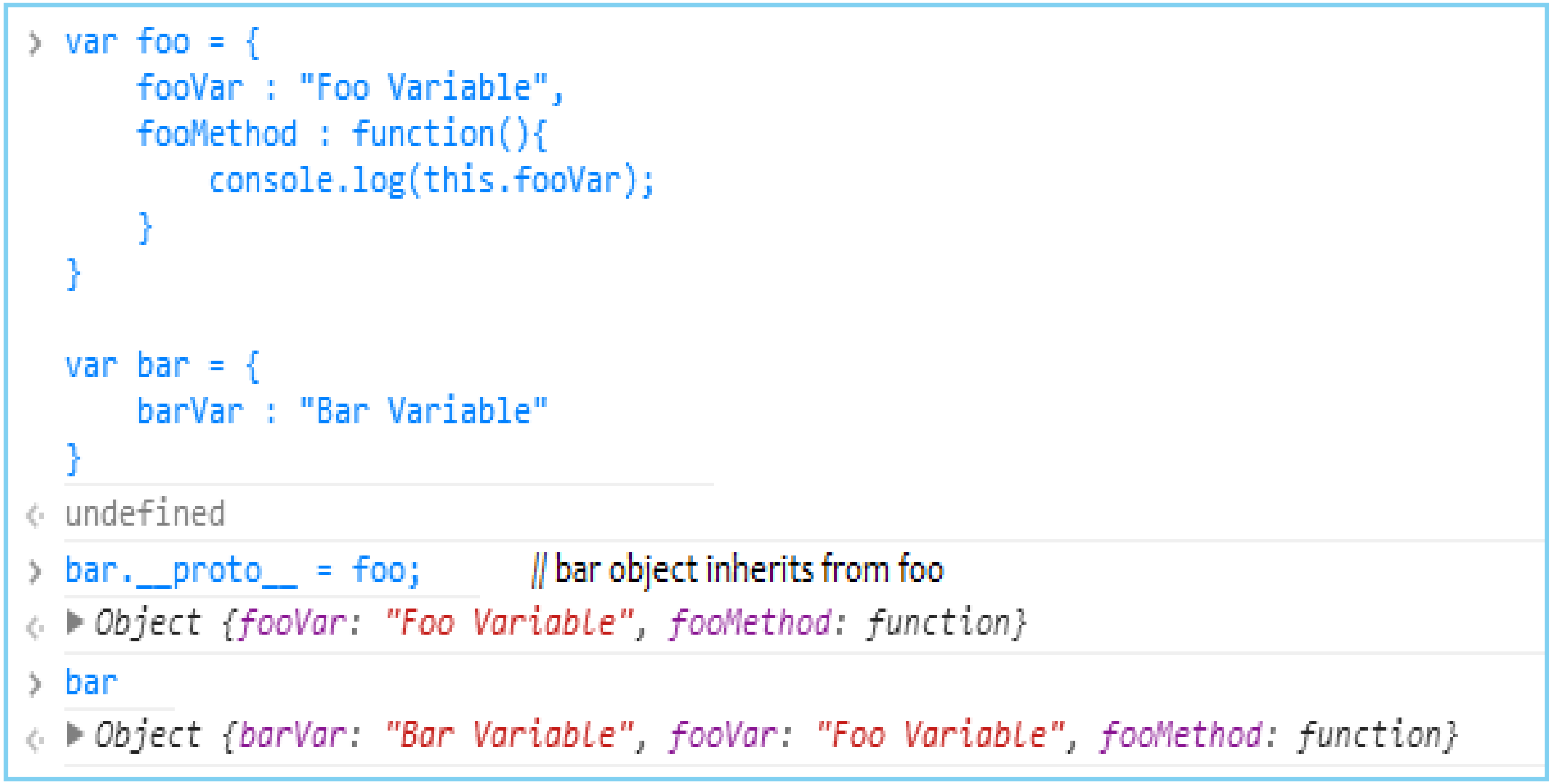
Agenda

- Prototypal inheritance
- Prototypal inheritance using __proto__
- Prototypal inheritance using create()
- Prototypal inheritance using prototype

Prototypal inheritance

- In JavaScript, the inheritance is prototype-based. Instead of class inherits from other class, an object inherits from another object.
- object inherits from another object using the following syntax.
- childObject.__proto__ = baseObject
  - Above mentioned syntax provided by Chrome / FireFox. In other browsers the property still exists internally, but it is hidden
- childObject = Object.create(baseObject)
- ConstructorFunction.prototype = baseObject
  - Above mentioned syntax works with all modern browsers.

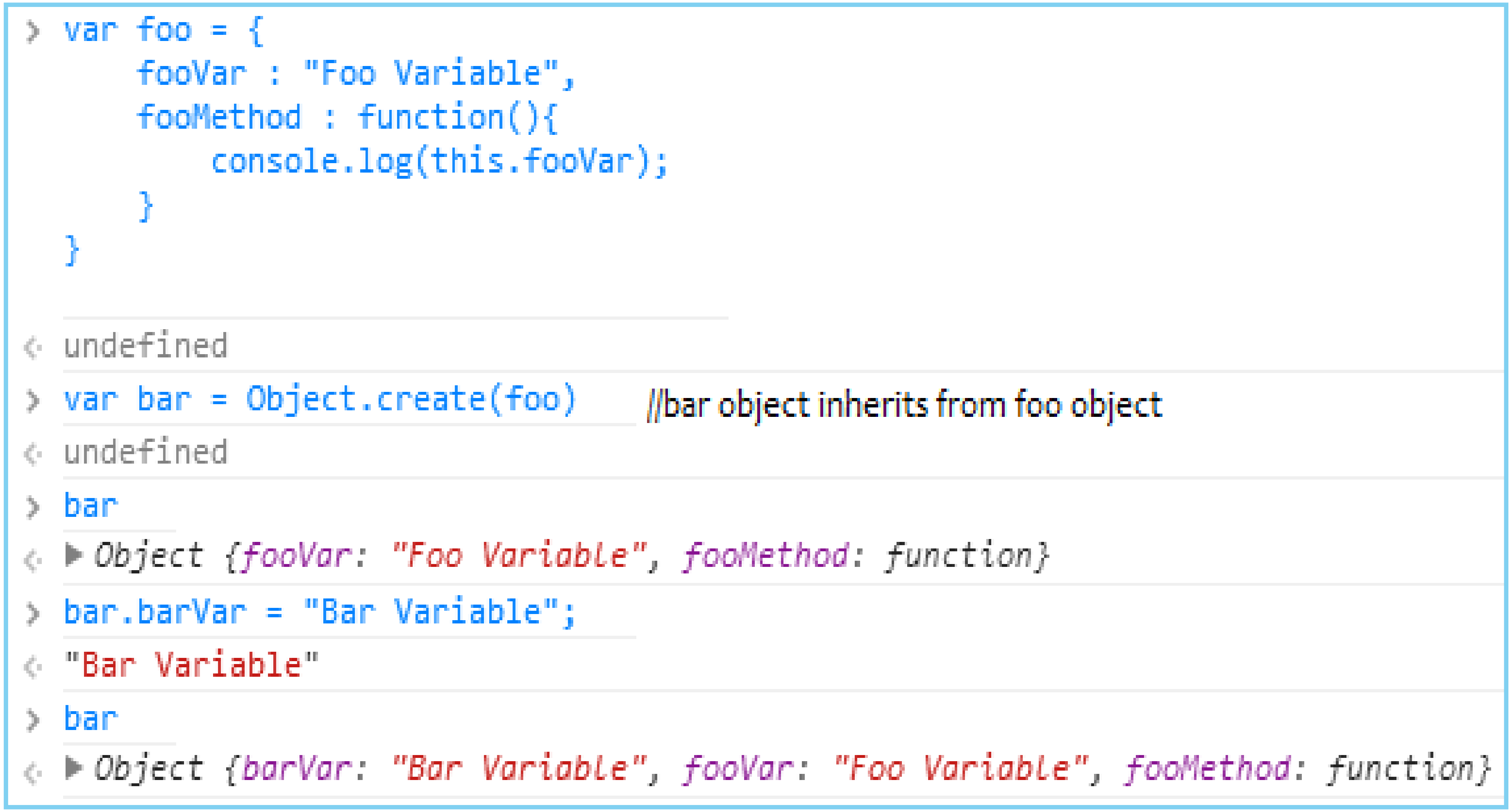
Prototypal inheritance using __proto__



```
> var foo = {
      fooVar : "Foo Variable",
      fooMethod : function(){
          console.log(this.fooVar);
      }
  }

  var bar = {
      barVar : "Bar Variable"
  }
< undefined
> bar.__proto__ = foo;        // bar object inherits from foo
< ▶ Object {fooVar: "Foo Variable", fooMethod: function}
> bar
< ▶ Object {barVar: "Bar Variable", fooVar: "Foo Variable", fooMethod: function}
```

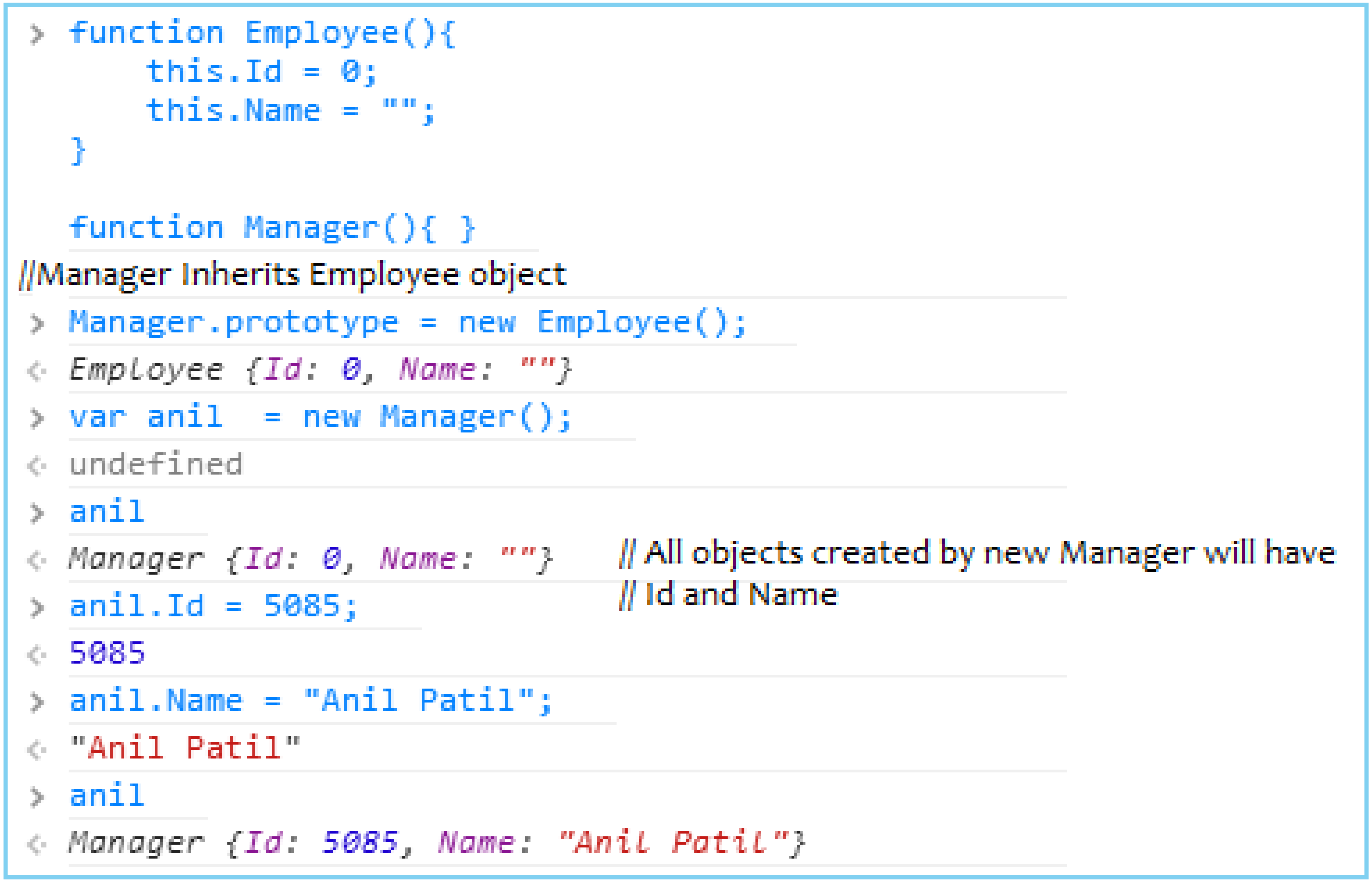
Prototypal inheritance using Object.create()



```
> var foo = {
      fooVar : "Foo Variable",
      fooMethod : function(){
          console.log(this.fooVar);
      }
  }

< undefined
> var bar = Object.create(foo)        //bar object inherits from foo object
< undefined
> bar
< ▶ Object {fooVar: "Foo Variable", fooMethod: function}
> bar.barVar = "Bar Variable";
< "Bar Variable"
> bar
< ▶ Object {barVar: "Bar Variable", fooVar: "Foo Variable", fooMethod: function}
```

Prototypal inheritance using prototype

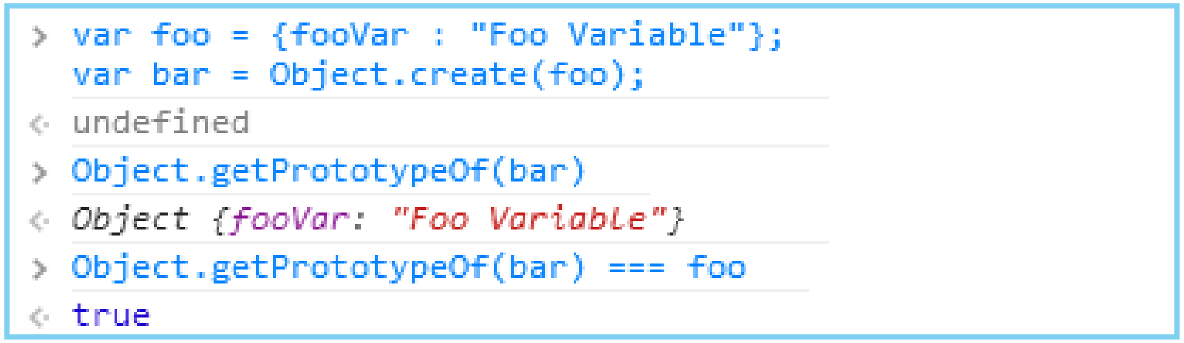


```
> function Employee(){
      this.Id = 0;
      this.Name = "";
  }

  function Manager(){ }
//Manager Inherits Employee object
> Manager.prototype = new Employee();
< Employee {Id: 0, Name: ""}
> var anil  = new Manager();
< undefined
> anil
< Manager {Id: 0, Name: ""}      // All objects created by new Manager will have
> anil.Id = 5085;                 // Id and Name
< 5085
> anil.Name = "Anil Patil";
< "Anil Patil"
> anil
< Manager {Id: 5085, Name: "Anil Patil"}
```

Prototypal inheritance

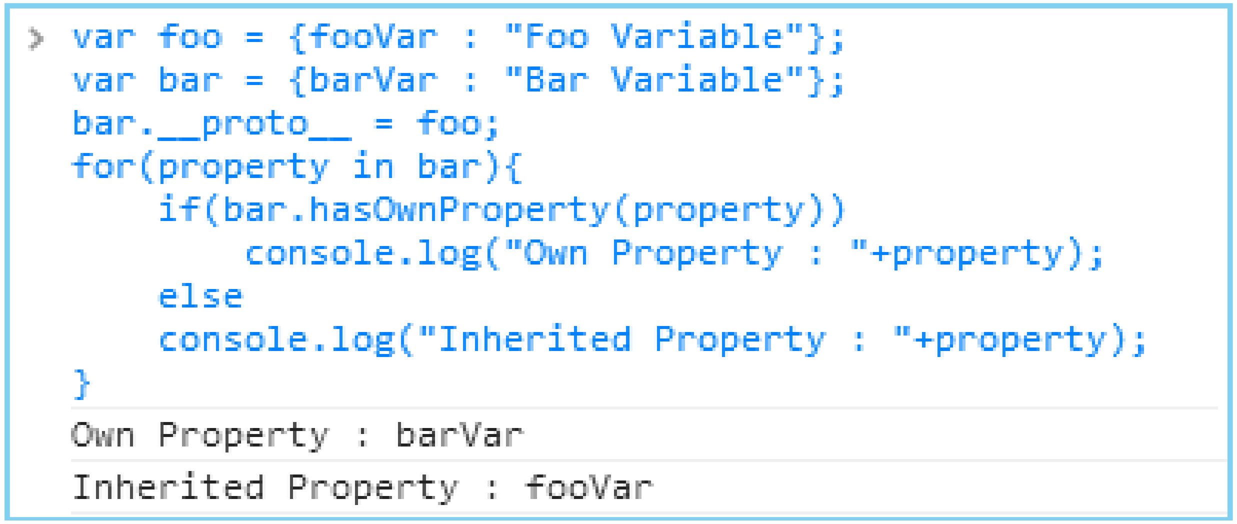
- Object.getPrototypeOf(obj) returns the value of obj.__proto__.



```
> var foo = {fooVar : "Foo Variable"};
  var bar = Object.create(foo);
< undefined
> Object.getPrototypeOf(bar)
< Object {fooVar: "Foo Variable"}
> Object.getPrototypeOf(bar) === foo
< true
```

- for..in loop lists properties in the object and its prototype chain.

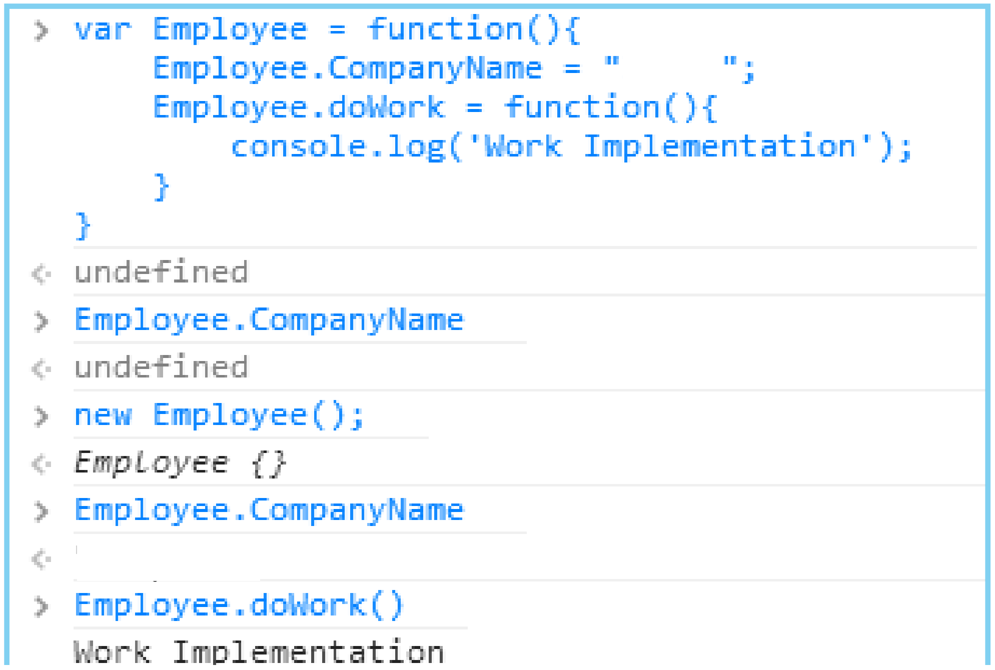
  obj.hasOwnProperty(prop) returns true if property belongs to that object.



```
> var foo = {fooVar : "Foo Variable"};
  var bar = {barVar : "Bar Variable"};
  bar.__proto__ = foo;
  for(property in bar){
      if(bar.hasOwnProperty(property))
          console.log("Own Property : "+property);
      else
      console.log("Inherited Property : "+property);
  }
  Own Property : barVar
  Inherited Property : fooVar
```

Static variables and methods

- In JavaScript we can directly put data into function object which acts like Static member.
- Static Members need to be accessed directly by Object name, cannot be accessed by reference variable. Static members gets created when the first object gets created.



```
> var Employee = function(){
      Employee.CompanyName = "       ";
      Employee.doWork = function(){
          console.log('Work Implementation');
      }
  }
< undefined
> Employee.CompanyName
< undefined
> new Employee();
< Employee {}
> Employee.CompanyName
<
> Employee.doWork()
  Work Implementation
```

Summary

- **In this lesson we have learned about -**
  - Prototypal inheritance
  - Prototypal inheritance using __proto__
  - Prototypal inheritance using create()
  - Prototypal inheritance using prototype