# Prototypal inheritance

* Every function has a property called prototype, where is methods and properties can be added to it (it is initially empty).
* When a function x creates x1, x1 will inherit the methods and properties in prototype.
* Javascript is unlike Java, where it is class-inheritance based.
* Though Javascript has the class keyword in ES6, it is still prototypal inheritance.

-----------------------------------------------------------

* For class based inheritance you have to inherit everything.

Every function expression is essentially a constructor in Javascript.

Below shows a simple example of how constructor is created with a function, and how objects can be instantiated from it:

var x = function() {

this.i = 0;

this.j = j;

this.getJ = function() {

return this.j;

}

}

var x1 = new x(1) ;

var x2 = new x(2);

x1 and x2 are instances of x, which means they are distinct objects. x1 and x2 inherits all the methods in x. So x is technically parent class. Every object will have the getJ method, which will take up space (unnecessarily) when the object is created.

A better solution will be to use “x.prototype.getJ = function() {}”. So that whenever getJ is called in x1 or x2 or x100 for that matter, the method will found in the parent (x) rather than from the individual objects.

