**OOPS IN JS**

#### JS supports OOP because it supports inheritance through prototyping as well as properties and methods. One of the most powerful features of the language is its flexibility. As a JavaScript programmer, you can make your programs as simple or as complex as you wish them to be.

#### Creating Objects:

1. Using new operator:

person = new Object()

person.name = "Tim Scarfe"

person.height = "6Ft"

person.run = function() {

this.state = "running"

this.speed = "4ms^-1"

}

1. Using Literal notation:

timObject = {

property1 : "Hello",

property2 : "MmmMMm",

property3 : ["mmm", 2, 3, 6, "kkk"],

method1 : function(){alert("Method had been called" + this.property1)}

};

timObject.method1();

alert(timObject.property3[2]) // will yield 3

var circle = { x : 0, y : 0, radius: 2 } // another example

// nesting is no problem.

var rectangle = {

upperLeft : { x : 2, y : 2 },

lowerRight : { x : 4, y : 4}

}

alert(rectangle.upperLeft.x) // will yield 2

#### Object constructor and prototyping:

1. **Create Class**

var mainClass = function(name1,city2){

this.name = name1;

this.city = city2;

};

1. **Add function to the class**

mainClass.**prototype**.getName = function(){ return this.name};

mainClass.**prototype**.getCity = function(){ return this.city};

/\* OR \*/

mainClass.**prototype** = {

getName:function(){ return this.name;},

getCity:function(){ return this.city;}

};

1. **Create the object of class and invoke services:**

var m = new mainClass("Pravin", "Malegaon");

alert(m.getName());

alert(m.getCity());

1. **We can assign new function to specific object**

m.getAge = function(){ return 30};

1. **Create Static function:**

mainClass.**getName** = function(){ return this.name};

**A Loosely Typed Language**

In JavaScript, you do not declare a type when defining a variable. However, this does not mean

that variables are not typed. Depending on what data it contains, a variable can have one of

several types.

There are three primitive types: booleans, numbers, and strings (JavaScript differs

from most other mainstream languages in that it treats integers and floats as the same type).

There are functions, which contain executable code. There are objects, which are composite

datatypes (an array is a specialized object, which contains an ordered collection of values).

Lastly, there are the null and undefined datatypes.

Primitive datatypes are passed by value,while all other datatypes are passed by reference. This can cause some unexpected side effects if you aren’t aware of it.

As in other loosely typed languages, a variable can change its type, depending on what

value is assigned to it. The primitive datatypes can also be cast from one type to another. The

toString method converts a number or boolean to a string. The parseFloat and parseInt functions convert strings to numbers. Double negation casts a string or a number to a boolean:

var bool = !!num;

**Anonymous function:**

(function() {

var foo = 10;

var bar = 2;

alert(foo \* bar);

})();

*In JavaScript, everything can be modified at run-time*