Example:\*/

function Rectangle(h,w) {

var width=w; // Both the 'width' and 'w' is private

var heigth=h; // Both the 'height' and 'h' is private

this.setWidth= function(w){ width=w;}

this.setHeight= function(h){ heigth=h;}

this.getWidth= function(){ return width;}

this.getHeight= function(){ return heigth;}

this.constructor.prototype.getDiagonal=function() {

return Math.sqrt(heigth\*heigth+width\*width);

};

}

Rectangle.prototype.getArea=function() {

// We must use accessors in a prototype kind of method,

// then these methods can not access the private members

// of a created object.

return this.getWidth()\*this.getHeight();

};

var rect = new Rectangle(60,70);

rect.setHeight(20);

document.write("The rectangle area is: "+rect.getArea()+"<br>");

document.write("The rectangle diagonal is: "+rect.getDiagonal()+"<br>");

/\*Private methods have no directly access to properties that are defined to be public with "this"

keyword in a function object. To achieve this, one can define a variable that has the same

reference as "this" refers to.

Example:\*/

function Person(n, y) {

var name=n;

var year=y;

// Set a variable to the same as this

var thisObj=this;

this.setName= function(n){ name=n;}

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

this.setYear= function(y){ year=y;}

this.getYear= function(){ return year;}

var born = function() {

var nYear=new Date().getFullYear();

// Use the thisObj variable which is the same as this refer to.

return nYear-thisObj.getYear();

// The "this" keyword inside this function refers to an object

// created by the constructor function of this function object.

}

this.getBornYear= function() {

return born();

}

}

var person = new Person("Nikita",60);

document.write(person.getName()+" was born in "+person.getBornYear()+"<br>");