Getter and Setter

```
// "Class"
function Point(){
        this.X = 0;
        this.Y = 0;
}
// Define getter
Object.__ def ine Get ter __.c al l(P oin t.p rot -
otype, " get Coo rds ", functi on(){
        return this.X + ", " + this.Y;
});
// Define setter
Object.__ def ine Set ter __.c al l(P oin t.p rot -
otype, " set Coo rds ", functi on( coo rds){
       var part = coords.to Str ing ().s pl -
it( ", ");
        this.X = part[0] || " ";
        this.Y = part[1] || " ";
});
// Usage
var testPoint = new Point();
testPo int.se tCoords = "10, 20";
docume nt.w ri te( tes tPo int.ge tCo ords);
```

Defining setter and getter

Inharitance

```
function Animal(){
        thi s.name = " Ani mal ";
        thi s.t oString = functi on(){
                return "This is " + this.name;
       };
}
function Canine(){
        thi s.name = " Can ine ";
}
function Wolf(){
        thi s.name = " Wol f";
}
// Inheriting
Canine.pr ototype = new Animal();
```

Inharitance (cont)

```
> Wolf.prototype = new Canine();
// There is need to re-reference constructors
Canine.prototype.constructor = Canine;
Wolf.prototype.constructor = Wolf;
// Function for quick inheriting
function extend(child, parent){
   var temp = function(){};
   temp.prototype = parent.prototype;
   child.prototype = new temp();
   child.prototype.constructor = child;
}
```

Defining get and set properties

```
function Point(){
        this.X = 0;
        this.Y = 0;
}
Object.de fin ePr ope rty (Po int.pr oto type,
'point Pos', {
       get: functi on(){
                return "X: " + this.X + " Y: " +
this.Y;
       }
       set: functi on( point){
               var parts = point.t oS tri -
ng( ).s pli t(", ");
                this.X = parts[0] || " ";
                this.Y = parts[1] || " ";
       }
});
// Usage
var testPoint = new Point();
testPo int.po intPos = "10, 20";
docume nt.w ri te( tes tPo int.po int Pos);
```


Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Define get and set prototype

```
function Circle(radius){
        thi s._ radius = radius;
}
// Define prototype
Circle.pr ototype = {
       set radius (ra dius){ this._ radius =
radius; }
       get radius(){ return this._ radius; }
       get area(){ return Math.PI * Math.p ow( -
thi s._ radius, 2); }
};
// Usage
var circle = new Circle (10);
circle.radius = 5;
docume nt.w ri te( cir cle.area);
// Usage
```

Call parent method

```
// Call parent method
Wolf.p rot oty pe.h eight = functi on(){
       var height = Canide.pr oto typ e.h eig -
ht.a pp ly( this);
        return height * 1.2;
};
```


Sponsored by **Readable.com**
Measure your website readability!
https://readable.com