JavaScript深入浅出

对象

@Bosn

概述

对象中包含一系列属性，这些属性是无序的。
每个属性都有一个字符串key和对应的value。

```
var obj = {x : 1, y : 2};
obj.x; // 1
obj.y; // 2
```

探索对象的key

```
var obj = {};
obj[1] = 1;
obj['1'] = 2;
obj; // Object {1: 2}

obj[{}] = true;
obj[{x : 1}] = true;
obj; // Object {1: 2, [object Object]: true}
```

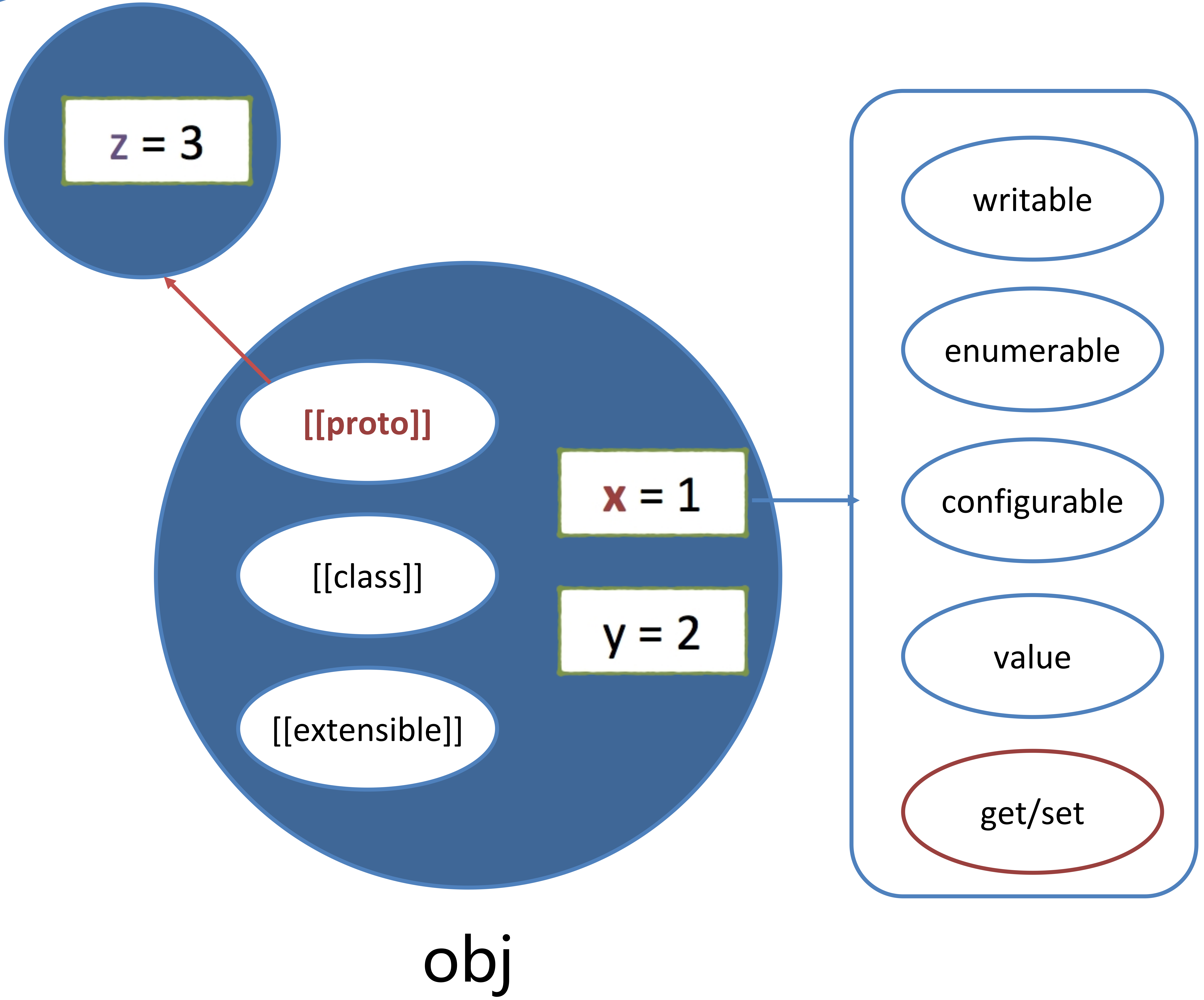
回顾-数据类型

object 对象
- Function
- Array
- Date
- ...

原始类型
- number
- string
- boolean
- null
- undefined

对象结构

```
var obj = {};
obj.y = 2;
obj.x = 1;


function foo(){}
foo.prototype.z = 3;
var obj =new foo();
```



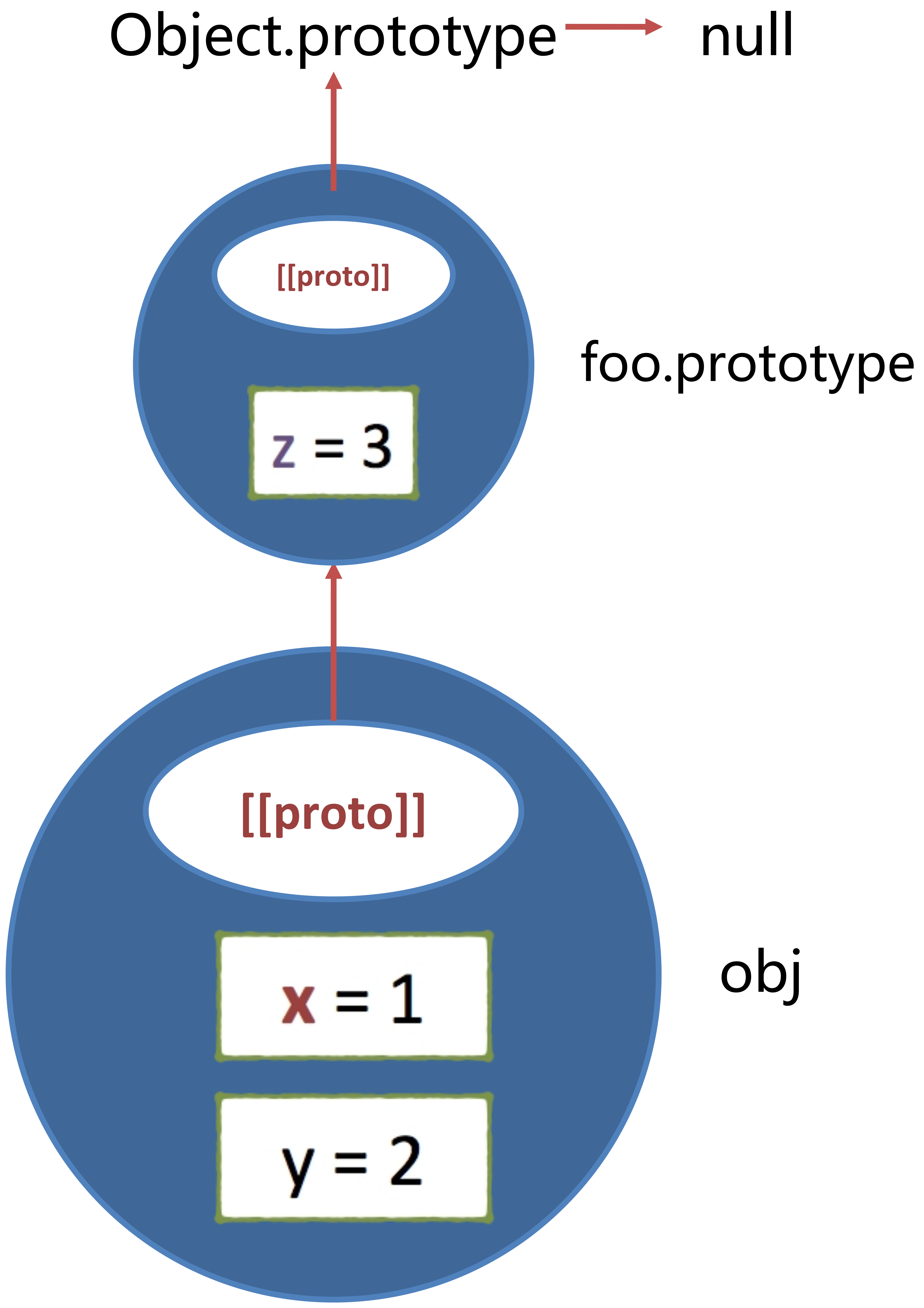
首先，你得有对象

对象创建-字面量

```
var obj1 = {x : 1, y : 2};

var obj2 = {
    x : 1,
    y : 2,
    o : {
        z : 3,
        n : 4
    }
};
```

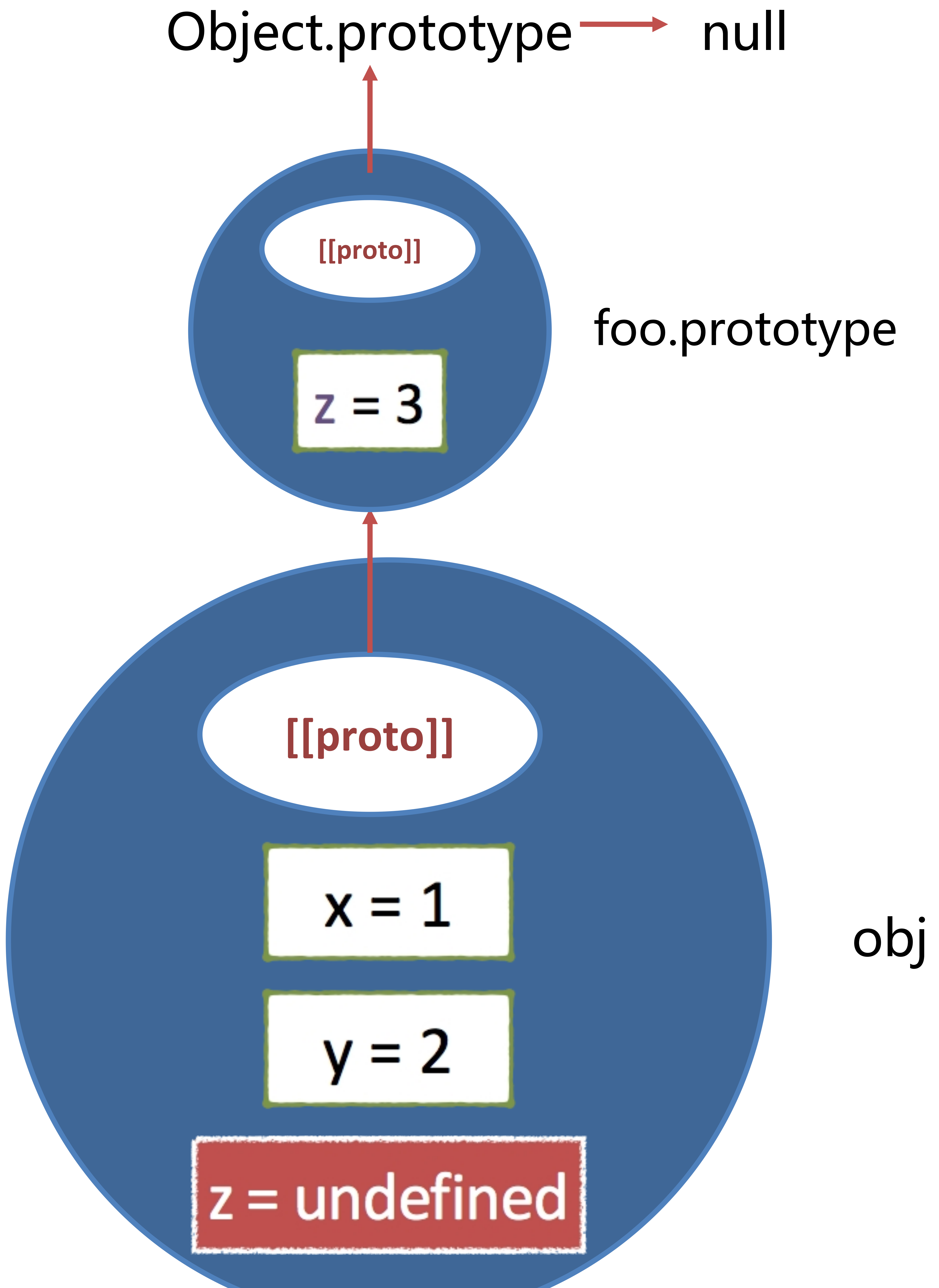
创建对象-new/原型链

```
function foo(){}
foo.prototype.z = 3;

var obj =new foo();
obj.y = 2;
obj.x = 1;

obj.x; // 1
obj.y; // 2
obj.z; // 3
typeof obj.toString; //   'function'
'z' in obj; // true
obj.hasOwnProperty('z'); // false
```



创建对象-new/原型链

```
obj.z = 5;

obj.hasOwnProperty('z'); // true
foo.prototype.z; // still 3
obj.z; // 5

obj.z = undefined;
obj.z; // undefined

delete obj.z; // true
obj.z; // 3

delete obj.z; // true
obj.z; // still 3!!!
```



对象创建-Object.create

```
var obj = Object.create({x : 1});
obj.x // 1
typeof obj.toString // "function"
obj.hasOwnProperty('x');// false
```

$$\begin{tikzcd}\text{obj} \arrow[r] & \{x:1\} \arrow[r] & \text{Object.prototype} \arrow[r] & \text{null}\end{tikzcd}$$

```
var obj = Object.create(null);
obj.toString // undefined
```

$$\begin{tikzcd}\text{obj} \arrow[r] & \text{null}\end{tikzcd}$$

属性操作

读写对象属性

属性异常

删除属性

检测属性

枚举属性

属性读写

```
var obj = {x : 1, y : 2};
obj.x; // 1
obj["y"]; // 2

obj["x"] = 3;
obj.y = 4;
```

```
var obj = {x1 : 1, x2 : 2};
var i = 1, n = 2;

for (; i <= n; i++) {
    console.log(obj['x' + i]);
}
// 输出: 1, 2

var p;
for (p in obj) {
    console.log(obj[p]);
}
```

属性读写-异常

```
var obj = {x : 1};
obj.y; // undefined
var yz = obj.y.z; // TypeError: Cannot read property 'z' of undefined
obj.y.z = 2; // TypeError: Cannot set property 'z' of undefined
```

```
var yz;
if (obj.y) {
    yz = obj.y.z;
}
```

```
var yz = obj && obj.y && obj.y.z;
```

属性删除

```
var person = {age : 28, title : 'fe'};
delete person.age; // true
delete person['title']; // true
person.age; // undefined
delete person.age; // true

delete Object.prototype; // false,

var descriptor = Object.getOwnPropertyDescriptor(Object, 'prototype');
descriptor.configurable; // false
```

属性删除

```
var globalVal = 1;
delete globalVal; // false

(function() {
    var localVal = 1;
    return delete localVal;
}()); // false
```

```
function fd() {}
delete fd; // false

(function() {
    function fd() {};
    return delete fd;
}()); // false
```

```
ohNo = 1;
window.ohNo; // 1
delete ohNo; // true
```

属性检测

```
var cat = new Object;
cat.legs = 4;
cat.name = "Kitty";

'legs' in cat; // true
'abc' in cat; // false
"toString" in cat; // true, inherited property!!!



cat.hasOwnProperty('legs'); // true
cat.hasOwnProperty('toString'); // false

cat.propertyIsEnumerable('legs'); // true
cat.propertyIsEnumerable('toString'); // false
```

属性检测

```
Object.defineProperty(cat, 'price', {enumerable : false, value : 1000});
cat.propertyIsEnumerable('price'); // false
cat.hasOwnProperty('price'); // true
```

```
if (cat && cat.legs) {
    cat.legs *= 2;
}
```

```
if (cat.legs != undefined) {
    // !== undefined, or, !== null
}
```

```
if (cat.legs !== undefined) {
    // only if cat.legs is not undefined
}
```

属性枚举

```
var o = {x : 1, y : 2, z : 3};
'toString' in o; // true
o.propertyIsEnumerable('toString'); // false
var key;
for (key in o) {
    console.log(key); // x, y, z
}
```

```
var obj = Object.create(o);
obj.a = 4;
var key;
for (key in obj) {
    console.log(key); // a, x, y, z
}
```

```
var obj = Object.create(o);
obj.a = 4;
var key;
for (key in obj) {
    if (obj.hasOwnProperty(key)) {
        console.log(key); // a
    }
}
```

另一种读写属性的方式

属性getter/setter方法

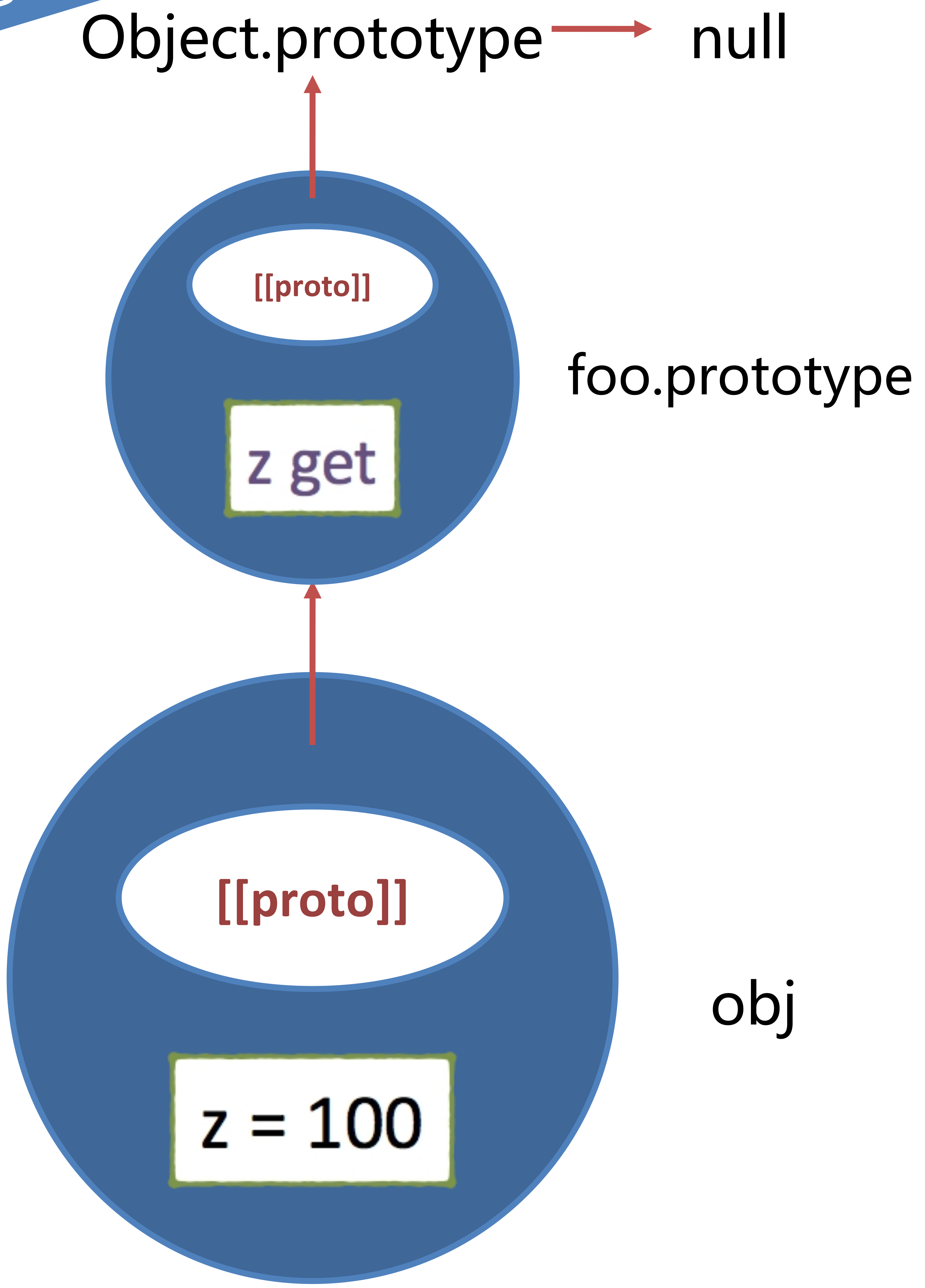
```
var man = {
    name : 'Bosn',
    weibo : '@Bosn',
    get age() {
        return new Date().getFullYear() - 1988;
    },
    set age(val) {
        console.log('Age can\'t be set to ' + val);
    }
}
console.log(man.age); // 27
man.age = 100; // Age can't be set to 100
console.log(man.age); // still 27
```

属性getter/setter方法

```
var man = {
    weibo : '@Bosn',
    $age : null,
    get age() {
        if (this.$age == undefined) {
            return new Date().getFullYear() - 1988;
        } else {
            return this.$age;
        }
    },
    set age(val) {
        val = +val;
        if (!isNaN(val) && val > 0 && val < 150) {
            this.$age = +val;
        } else {
            throw new Error('Incorrect val = ' + val);
        }
    }
}
```

```
console.log(man.age); // 27
man.age = 100;
console.log(man.age); // 100;
man.age = 'abc'; // error:Incorrect val = NaN
```

get/set与原型链



```
function foo() {}

Object.defineProperty(foo.prototype, 'z',
   {get : function(){return 1;}});

var obj = new foo();

obj.z; // 1
obj.z = 10;
obj.z; // still 1
```

```
Object.defineProperty(obj, 'z',
{value : 100, configurable: true});
obj.z; // 100;
delete obj.z;
obj.z; // back to 1
```

get/set与原型链

```
var o = {};
Object.defineProperty(o, 'x', {value : 1}); // writable=false, configurable=false
var obj = Object.create(o);
obj.x; // 1
obj.x = 200;
obj.x; // still 1, can't change it

Object.defineProperty(obj, 'x', {writable:true, configurable:true, value : 100});
obj.x; // 100
obj.x = 500;
obj.x; // 500
```

属性级的权限设置

属性标签

```
Object.getOwnPropertyDescriptor({pro : true}, 'pro');
// Object {value: true, writable: true, enumerable: true, configurable: true}
Object.getOwnPropertyDescriptor({pro : true}, 'a'); // undefined
```

```
var person = {};
Object.defineProperty(person, 'name', {
    configurable : false,
    writable : false,
    enumerable : true,
    value : "Bosn Ma"
});
```

```
person.name; // Bosn Ma
person.name = 1;
person.name; // still Bosn Ma
delete person.name; // false
```

属性标签

```
Object.defineProperty(person, 'type', {
    configurable : true,
    writable : true,
    enumerable : false,
    value : "Object"
});

Object.keys(person); // ["name"]
```

属性标签

```
Object.defineProperties(person, {
    title : {value : 'fe', enumerable : true},
    corp : {value : 'BABA', enumerable : true},
    salary : {value : 50000, enumerable : true, writable : true}
});

Object.getOwnPropertyDescriptor(person, 'salary');
// Object {value: 50000, writable: true, enumerable: true, configurable: false}
Object.getOwnPropertyDescriptor(person, 'corp');
// Object {value: "BABA", writable: false, enumerable: true, configurable: false}
```

属性标签

```
Object.defineProperties(person, {
    title : {value : 'fe', enumerable : true},
    corp : {value : 'BABA', enumerable : true},
    salary : {value : 50000, enumerable : true, writable : true},
    luck : {
        get : function() {
        return Math.random() > 0.5 ? 'good' : 'bad';
        }
    },
    promote : {
        set : function (level) {
            this.salary *= 1 + level * 0.1;
        }
    }
});

Object.getOwnPropertyDescriptor(person, 'salary');
// Object {value: 50000, writable: true, enumerable: true, configurable: false}
Object.getOwnPropertyDescriptor(person, 'corp');
// Object {value: "BABA", writable: false, enumerable: true, configurable: false}
person.salary; // 50000
person.promote = 2;
person.salary; // 60000
```

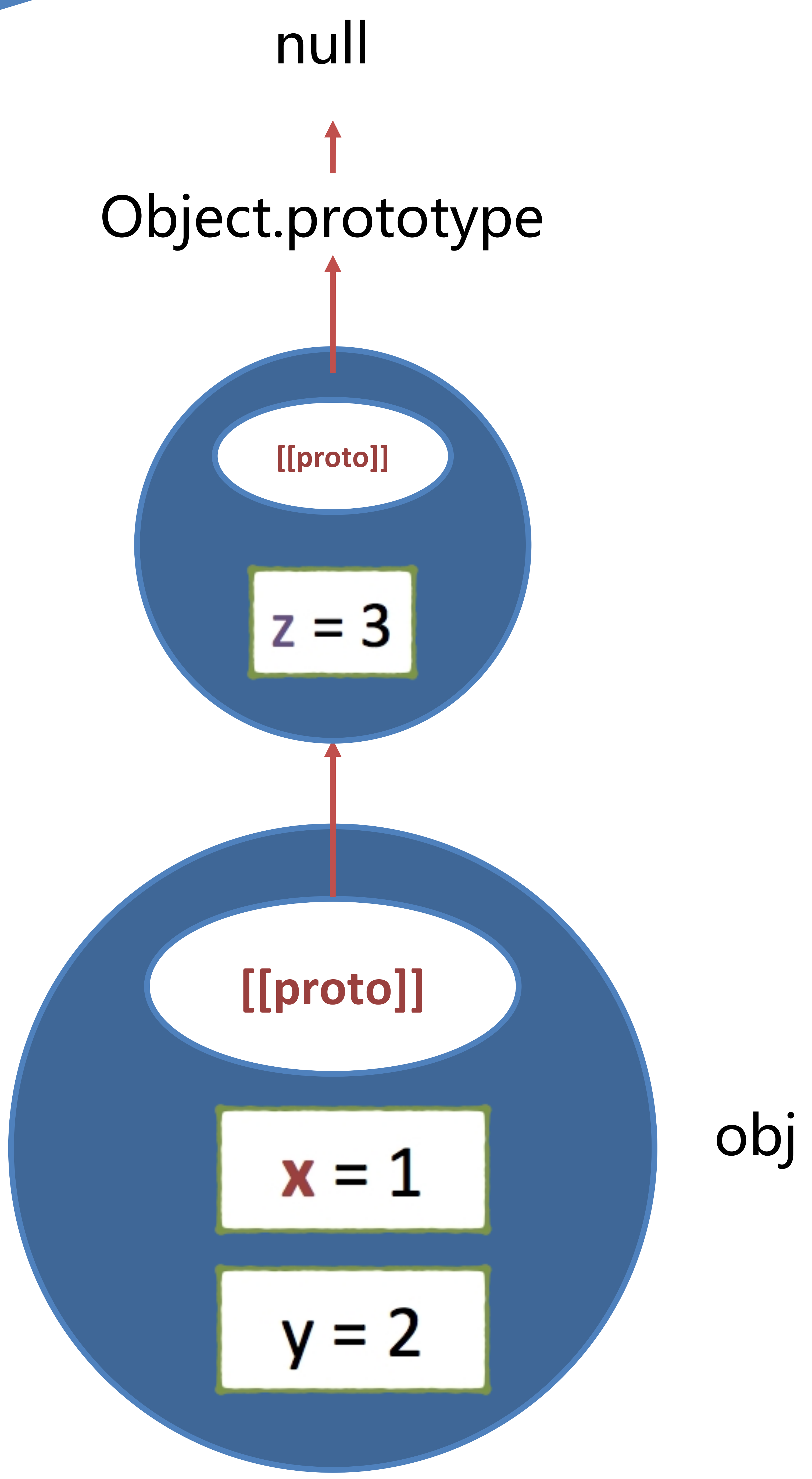
属性标签

	configurable:true writable:true	configurable:true writable:false	configurable:false writable:true	configurable:false writable:false
修改属性的值	√	√* 重设value标签修改	√	✗
通过属性赋值　修改属性的值	√	✗	√	✗
delete该属性返回true	√	√	✗	✗
修改getter/setter方法	√	√	✗	✗
修改属性标签* (除了writable从true修改为false总是允许)	√	√	✗	✗

对象标签

[[proto]]
[[class]]
[[extensible]]

原型标签__proto__



class标签

```
var toString = Object.prototype.toString;
function getType(o){return toString.call(o).slice(8,-1);};

toString.call(null); // "[object Null]"
getType(null); // "Null"
getType(undefined); // "Undefined"
getType(1); // "Number"
getType(new Number(1)); // "Number"
typeof new Number(1); // "object"
getType(true); // "Boolean"
getType(new Boolean(true)); // "Boolean"
```

extensible标签

```
var obj = {x : 1, y : 2};
Object.isExtensible(obj); // true
Object.preventExtensions(obj);
Object.isExtensible(obj); // false
obj.z = 1;
obj.z; // undefined, add new property failed
Object.getOwnPropertyDescriptor(obj, 'x');
// Object {value: 1, writable: true, enumerable: true, configurable: true}

Object.seal(obj);
Object.getOwnPropertyDescriptor(obj, 'x');
// Object {value: 1, writable: true, enumerable: true, configurable: false}
Object.isSealed(obj); // true

Object.freeze(obj);
Object.getOwnPropertyDescriptor(obj, 'x');
// Object {value: 1, writable: false, enumerable: true, configurable: false}
Object.isFrozen(obj); // true

// [caution] not affects prototype chain!!!
```

序列化、其它对象方法

序列化

```
var obj = {x : 1, y : true, z : [1, 2, 3], nullVal : null};
JSON.stringify(obj); // "{"x":1,"y":true,"z":[1,2,3],"nullVal":null}"

obj = {val : undefined, a : NaN, b : Infinity, c : new Date()};
JSON.stringify(obj); // "{"a":null,"b":null,"c":"2015-01-20T14:15:43.910Z"}"

obj = JSON.parse('{"x" : 1}');
obj.x; // 1
```

序列化 - 自定义

```
var obj = {
    x : 1,
    y : 2,
    o : {
        o1 : 1,
        o2 : 2,
        toJSON : function () {
            return this.o1 + this.o2;
        }
    }
};
JSON.stringify(obj); // "{"x":1,"y":2,"o":3}"
```

其它对象方法

```
var obj = {x : 1, y : 2};
obj.toString(); // "[object Object]"
obj.toString = function() {return this.x + this.y};
"Result " + obj; // "Result 3", by toString

+obj; // 3, from toString

obj.valueOf = function() {return this.x + this.y + 100;};
+obj; // 103, from valueOf

"Result " + obj; // still "Result 3"
```

小结

- 对象的结构
- 创建对象
- 属性操作
- getter setter
- 属性标签
- 对象标签
- 序列化
- 对象方法

谢谢