面向对象的JavaScript

对象属性
创建对象
继承

对象属性 – 一般方法

```
person.a = 1;
person[“a”] = 1;
person = {a: 1};
```

对象属性 – 数据属性

```
var person = {};
Object.defineProperty(person, "name", {
    writable: false,
    value: "Nicholas"
});

alert(person.name);    //"Nicholas"
person.name = "Greg";
alert(person.name);    //"Nicholas"
```

configurable

enumerable

writable

value

对象属性 – 访问器属性

```
var book = {
    _year: 2004,
    edition: 1
};

Object.defineProperty(book, "year", {
    get: function(){
        return this._year;
    },
    set: function(newValue){
        if (newValue > 2004) {
            this._year = newValue;
            this.edition += newValue - 2004;
        }
    }
});

book.year = 2005;
alert(book.edition);   //2
```

configurable

enumerable

get

set

创建对象 – 工厂模式

```
function createPerson(name, age, job){
    var o = new Object();
    o.name = name;
    o.age = age;
    o.job = job;
    o.sayName = function(){
        alert(this.name);
    };
    return o;
}

var person1 = createPerson("Nicholas", 29, "Software Engineer");
var person2 = createPerson("Greg", 27, "Doctor");
```

创建 – 构造函数

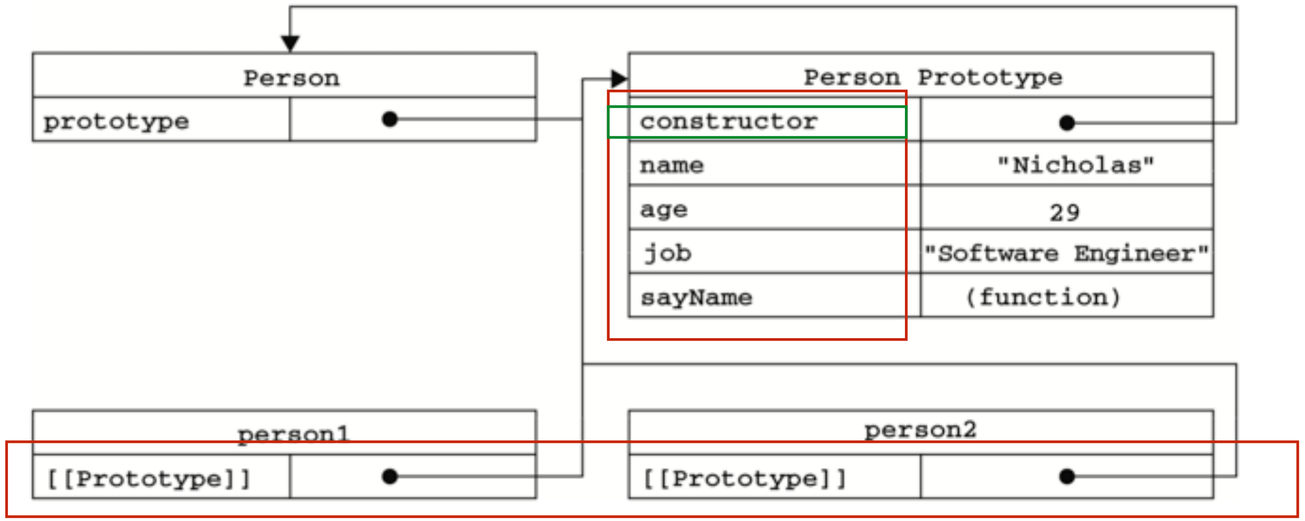
```
function Person(name, age, job){
    this.name = name;
    this.age = age;
    this.job = job;
    this.sayName = function(){
        alert(this.name);
    };
}

var person1 = new Person("Nicholas", 29, "Software Engineer");
var person2 = new Person("Greg", 27, "Doctor");
```

```
> function a() { return 1; } new a() instanceof a
< true
> function a() { return {}; } new a() instanceof a
< false
> function a() { } new a() instanceof a
< true
```

创建对象 – 原型模式

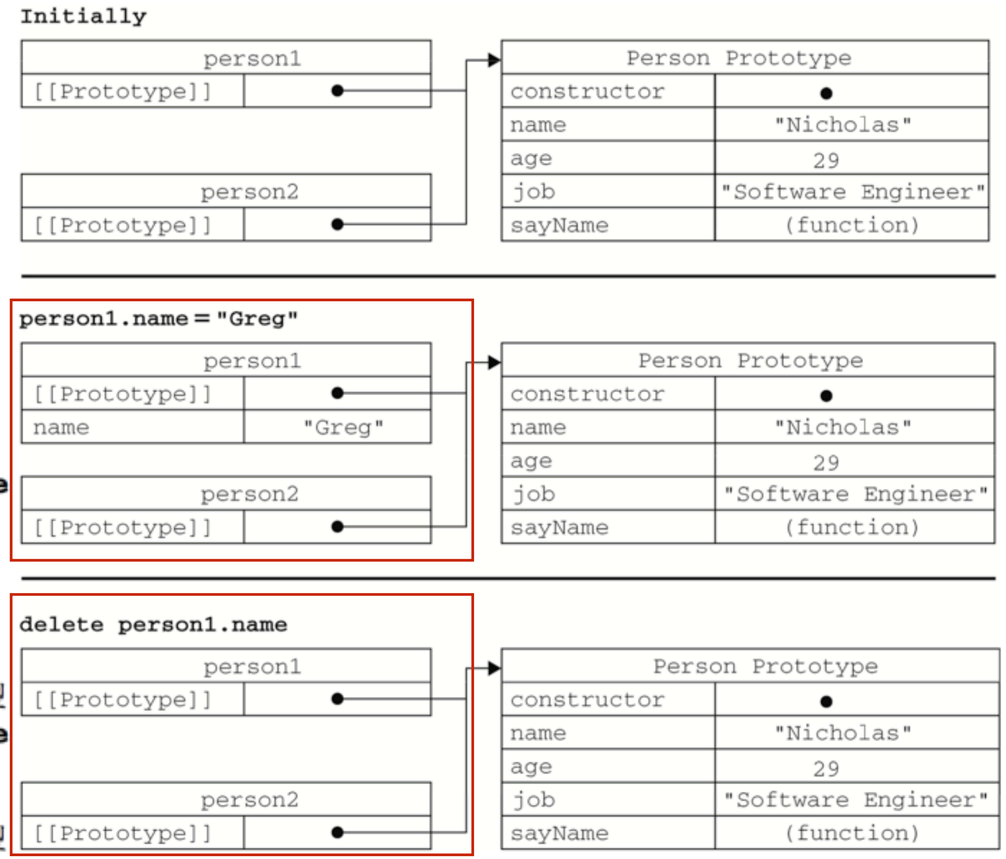
```
function Person(){
}

Person.prototype.name = "Nicholas";
Person.prototype.age = 29;
Person.prototype.job = "Software Engineer";
Person.prototype.sayName = function(){
    alert(this.name);
};

var person1 = new Person();
person1.sayName();   //"Nicholas"

var person2 = new Person();


person2.sayName();   //"Nicholas"

alert(person1.sayName == person2.sayName);  //true
```

创建对象 – 原型模式


Person
prototype
Person Prototype
constructor
name
"Nicholas"
age
29
job
"Software Engineer"
sayName
(function)
person1
[[Prototype]]
person2
[[Prototype]]

创建对象 – 实例属性和原型属性

```
function Person(){
}

Person.prototype.name = "Nicholas";
Person.prototype.age = 29;
Person.prototype.job = "Software Engineer";
Person.prototype.sayName = function(){
    alert(this.name);
};

var person1 = new Person();
var person2 = new Person();

alert(person1.hasOwnProperty("name"));  //false

person1.name = "Greg";
alert(person1.name);      //"Greg"——来自实例
alert(person1.hasOwnProperty("name"));  //true

alert(person2.name);      //"Nicholas"——来自原型
alert(person2.hasOwnProperty("name"));  //false

delete person1.name;
alert(person1.name);      //"Nicholas"——来自原型
alert(person1.hasOwnProperty("name"));  //false
```



hasOwnProperty

in

```
function Person(){
}

Person.prototype.name = "Nicholas";
Person.prototype.age = 29;
Person.prototype.job = "Software Engineer";
Person.prototype.sayName = function(){
    alert(this.name);
};

var person1 = new Person();
var person2 = new Person();

alert(person1.hasOwnProperty("name"));  //false
alert("name" in person1);  //true

person1.name = "Greg";
alert(person1.name);   //"Greg" ——来自实例
alert(person1.hasOwnProperty("name"));  //true
alert("name" in person1);  //true

alert(person2.name);   //"Nicholas" ——来自原型
alert(person2.hasOwnProperty("name"));  //false
alert("name" in person2);  //true

delete person1.name;
alert(person1.name);   //"Nicholas" ——来自原型
alert(person1.hasOwnProperty("name"));  //false
alert("name" in person1);  //true
```

for in

创建对象 – 组合

构造函数的问题
原型的问题

创建对象 – 组合

```
function Person(name, age, job){
    this.name = name;
    this.age = age;
    this.job = job;
    this.friends = ["Shelby", "Court"];
}

Person.prototype = {
    constructor : Person,
    sayName : function(){
        alert(this.name);
    }
}

var person1 = new Person("Nicholas", 29, "Software Engineer");
var person2 = new Person("Greg", 27, "Doctor");

person1.friends.push("Van");
alert(person1.friends);    //"Shelby,Count,Van"
alert(person2.friends);    //"Shelby,Count"
alert(person1.friends === person2.friends);    //false
alert(person1.sayName === person2.sayName);    //true
```

继承