什么是面向对象：

对象是一个实例、物体可以指代活的动物也可以指代死的物体。它有自己的属性，比如说颜色、重量、宽度、高度等等

又可以供外界操作的方法，比如说拿起来、放下去、奔跑、大喊等等那里应用

在程序中面向对象简单理解根据定义好的类，类就是抽象化的对象，可以理解成这个对象的类别，对象是类的一种具体实现。在编程中我们定义一个类，里面有属性，有方法。然后调用的时候就是如java，对象 = new 类，对象.方法。

具体的实现我也是一知半解，懂一些又不懂一些，就不深入了解了

在js要实现面向对象编程的话需要用到 this、prototype、new

this:在妙味课堂2011年的js讲解视频中，this表示当前方法属于谁，而另一种说法认为this是哪里是谁调用的就指向谁，比如说，

Init（）；//window调用

function init（）

{

alert（this）；//这里的this是window，因为是在全局作用域下直接调用的，

//在全局作用域下自然都是window调用的，笔者在chrome下测试过

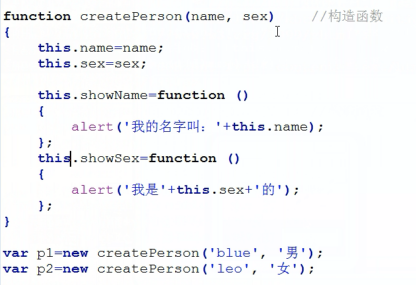
}

但是这只在非严格模式中才有效

这种理解来自《你不知道的JavaScript》一书中关于this的讲解

笔者比较倾向于第一种

还有一些例子是



来自妙味课堂2011年js视频教程

这里是new 调用了createPerson方法，createPerson方法里有this，这里this指向什么呢，笔者认为是指向new出来的对象

在妙味课堂2011js视频教程里是这样的



讲师说假想（估计是为了让我们好理解）function里面js帮你new出来一个对象，然后存放在叫this的变量里，然后同时帮我们return回去

Prototype（原型）： 理解来自妙味课堂2011年js视频教程

Prototype就相当于css样式中的class，给一类节点元素加上相同的样式

在js，prototype就是给一个类（这个类是个人理解，我认为在js中类声明一个类等于一个写了构造函数）加方法，用这个类实例化的对象都有这个方法

用法：

给js内置类加方法：Array.prototype.sum = function (){}//给js的Array类加sum方法

var arr1 = new Array(12,12,12,5);var arr2 = new Array(4,6);

arr1.sum()//41

arr2.sum()//10

给自己声明的类添加方法：

function person(name , sex)//是一个类，也是一个构造方法

{

this.name = name;

this.sex = sex;

}

person.prototype.showName() = function()

{

alert(“我的名字是” + this.name);  
}

person.prototype.showSex() = function()

{

alert(“我的名字是” + this.sex);  
}

var p1 = new person(“xx”,”nv”);

var p2 = new person(“yy”,”nan”);

p1.showName()

p2.showSex()

p1.showSex()

p2.showName()

注：构造函数（类）的首字母大写，一种编程规范

相同的东西加到原型里，不同的东西加到构造函数（类）中

原型的优先级

原型的优先级是要低一点的，var arr = new Array(); Array.prototype.a = 10;arr.a = 15 //优先级高

相当与arr.a =15 是行间样式，Array.prototype.a = 10是class

Json式面向对象

var p1=

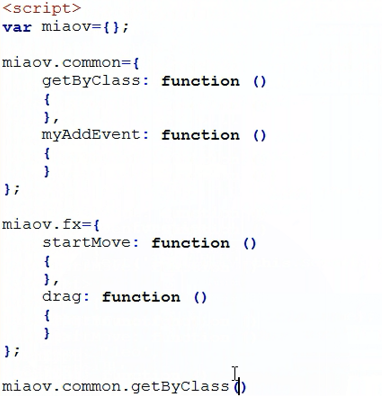
{

name : “blue”,

sex:”nan”,

showName:function(){alert(this.name);},

showSex:function(){alert(this.sex);}  
}这种一般用作命名空间



适用于用来写工具

////////////////////////////////////////////////////////////////////////////////////

Js的继承： 需要用到js的call函数

function Person(name , sex)//是一个类，也是一个构造方法

{

this.name = name;

this.sex = sex;

}

person.prototype.showName() = function()

{

alert(“我的名字是” + this.name);  
}

person.prototype.showSex() = function()

{

alert(“我的名字是” + this.sex);  
}

////////////////////////////////////////////////////////////////////////

function Worker(name,sex,job)

{

//调用父级的构造函数

//在这里面this指向new出来的worker对象，call函数可以

//把this的指向改变，当Person（）；这样子改成Person.call（this，name，sex）

//Person构造函数里的this就会变成new出来的worker对象

//这样子类就会继承父类的属性

Person.call(this,name,sex);

this.job = job;

}

//将父级的原型给子类的原型，叫原型链

//这样子写的话即Worker原型指向和Person原型指向为同一个东西

//这样子的话父类或子类更改东西的话会影响到其他继承的类。

//这种写法就不好

//Worker.prototype = Person.prototype;

第二种继承：

for（var i in Person.prototype）

{

Worker.prototype[i] = Person.prototype[i];

}

Woker.showJob()

{

alert(“job is ” + this.job);

{