**理解原型**

原型是一个对象，其他对象可以通过它实现属性继承。任何一个对象都有一个prototype的属性，记为:\_\_proto\_\_。每当我们定义一个对象，其\_\_proto\_\_属性就指向了其prototype。示例如下：

var foo = {

x: 10,

y: 20

};

**使用原型**

理解了原型的原理之后，如何使用原型呢？或者说原型有什么作用呢？

一般的初学者，在刚刚学习了基本的javascript语法后，都是通过面向函数来编程的。如下代码：

var decimalDigits = 2, tax = 5;

function add(x, y) {

return x + y;

}

function subtract(x, y) {

return x - y;

}

通过执行各个函数来得到最后的结果。但是利用原型，我们可以优化一些我们的代码，使用构造函数：

首先，函数本体中只存放变量：

var Calculator = function (decimalDigits, tax) {

this.decimalDigits = decimalDigits;

this.tax = tax;

};

其具体的方法通过prototype属性来设置：

Calculator.prototype = {

add: function (x, y) {

return x + y;

},

subtract: function (x, y) {

return x - y;

}

};

alert((new Calculator()).add(1, 3));

这样就可以通过实例化对象后进行相应的函数操作。这也是一般的js框架采用的方法。

原型还有一个作用就是用来实现继承。首先，定义父对象：

var BaseCalculator = function() {

this.decimalDigits = 2;

};

BaseCalculator.prototype = {

add: function(x, y) {

return x + y;

},

subtract: function(x, y) {

return x - y;

}

};

然后定义子对象，将子对象的原型指向父元素的实例化：

var Calculator = function () {

//为每个实例都声明一个税收数字

this.tax = 5;

};

Calculator.prototype = new BaseCalculator();

我们可以看到Calculator的原型是指向到BaseCalculator的一个实例上，目的是让Calculator集成它的add(x,y)和subtract(x,y)这2个function，还有一点要说的是，由于它的原型是BaseCalculator的一个实例，所以不管你创建多少个Calculator对象实例，他们的原型指向的都是同一个实例。

如果我不想让Calculator访问BaseCalculator的构造函数里声明的属性值，那怎么办呢？只需要将Calculator指向BaseCalculator的原型而不是实例就行了。代码如下：

var Calculator = function () {

this.tax= 5;

};

Calculator.prototype = BaseCalculator.prototype;

在使用第三方库的时候，有时候他们定义的原型方法不能满足我们的需要，我们就可以自己添加一些方法，代码如下：

//覆盖前面Calculator的add() function

Calculator.prototype.add = function (x, y) {

return x + y + this.tax;

};

var calc = new Calculator();

alert(calc.add(1, 1));

**原型链**

对象的原型指向对象的父，而父的原型又指向父的父，这种原型层层的关系，叫做原型链。

在查找一个对象的属性时，javascript会向上遍历原型链，直到找到给定名称的属性为止，当查找到达原型链的顶部，也即是Object.prototype，仍然没有找到指定的属性，就会返回undefined。

**ES6继承与原型**

class A{

constructor(param){

}

add(){}

}

class B extends A{

constructor(param){

super(param)

}

}

// 经过编译的B.prototype如下

// prototype是一个包含了类A的所有方法的对象，所有继承A类的类，其原型都是使用同一个对象

prototype = {

constructor: function(param){

},

add: function(){},

prototype: {}

}