# 概述

JS是一种基于对象的语言，对象的概念在JS体系中十分的重要，因此有必要清楚地了解一下JS中对象创建的常用方法及各自的局限性。

JavaScript中声明对象的简单符号是 “{}”。

# 使用Object或对象字面量创建对象（new）

最基础的创建对象方法

## 基本方法：

var student = new Object();

student.name = "easy";

student.age = "20";

## 对象字面量创建

var sutdent = {

name : "easy",

age : 20

};

# 工厂模式

1. 调用函数，输入参数，平批量创建对象（与上一种没有本质区别，只是增加了代码的复用性）
2. 返回一律是Object类型，无法区分对象类型。

function createStudent(name, age) {

var obj = new Object();

obj.name = name;

obj.age = age;

//返回对象

return obj;

}

var student1 = createStudent("easy1", 20);

var student2 = createStudent("easy2", 20);

...

var studentn = createStudent("easyn", 20);

# 构造函数创建对象

创建对象会占用大量内存，主要是因为每一个对象都不一样。

## 说明：

1. 实际上并不存在创建构造函数的特殊语法。
2. 构造函数与普通函数唯一的区别在于调用方法。
3. 对于任意函数，使用new操作符调用，那么它就是**构造函数**；
4. 对于任意函数，不使用new操作符调用，那么它就是**普通函数**。
5. 按照惯例，我们约定**构造函数名**以大写字母开头，**普通函数**以小写字母开头，
6. 使用new操作符调用构造函数时，会经历
7. 创建一个新对象；
8. 将构造函数作用域赋给新对象（使this指向该新对象）；
9. 执行构造函数代码；
10. 返回新对象；

## 例子：

1. 普通函数。

function student(name, age) {

this.name = name;

}

//调用

student(“yi”, 19);

1. 构造函数

function Student(name, age) {

this.name = name;

}

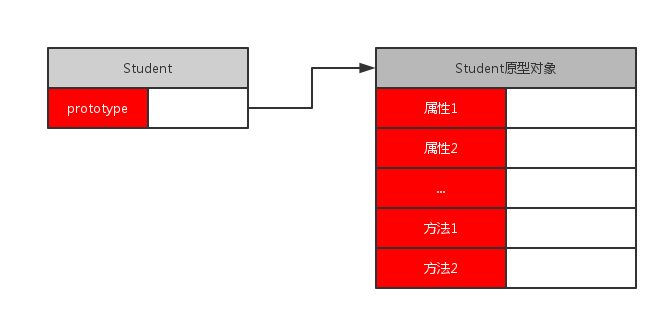
//调用

var v1 = new Student("easy", 20);

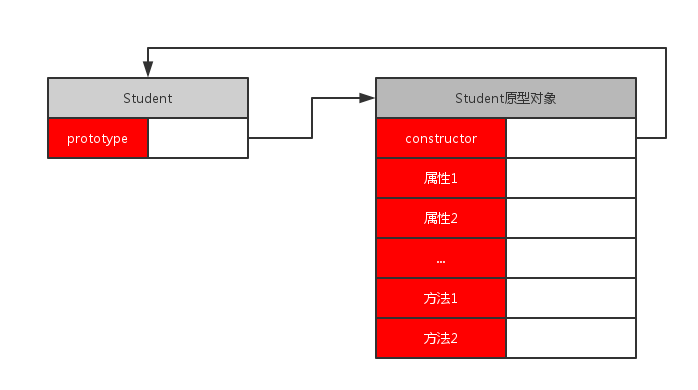
# 原型模式

## 说明：

我们创建的每一个函数都有一个prototype属性，该属性是一个指针，该指针指向了一个对象。对于我们创建的构造函数，该对象中包含可以由所有实例共享的属性和方法。如下如所示：



在默认情况下，所有原型对象会自动包含一个constructor属性，该属性也是一个指针，指向prototype所在的函数：



## 例子：

function Student() {

//定义Student类特有的成员属性

Var a;

//定义共有变量或方法->也可写到函数外

Student.prototype.name = "easy";

Student.prototype.age = 20;

Student.prototype.alertName = function(){

alert(this.name);

};

}

//调用

var stu1 = new Student();

Student.name;