# JS模式

javascript灵活性

创建类1 类

var Cls\_Anim = function(){

}

//把方法赋给 prototype属性

Cls\_Anim.prototype.start = function(){

}

Cls\_Anim.prototype.stop = function(){

}

var obj\_Anim = new Cls\_Anim();

obj\_Anim.start();

obj\_Anim.stop();

创建类2 类的定义封装在在一条声明中：

var Cls\_Anim = function(){

}

Cls\_Anim.prototype = {

start:function(){

},

stop:function(){

}

}

创建类3

//为类添加新方法 参数（方法的名称，方法内容）

Funtion.prototype.method = function(name , fn){

this.prototype[name] = fn;

}

var Cls\_Anim = function(){

}

Cls\_Anim.method('start',function(){

});

Cls\_Anim.method('stop',function(){

});

弱类型语言

1 定义变量不必声明类型

2 变量有原始类型（布尔，数值，字符串），对象类型，函数类型，空类型(null)，未定义类型（undefined）

3 原始数据类型按值传送，其他按照引用传值

4 类型转换，toString()把数值和布尔值转换为字符串，parseFloat parseInt 字符串转换成数值

函数是一等对象

1 存储在变量中 作为参数传递

2 匿名函数 function(){}

(funtion(){

})()

(funtion(a,b){

})(10,2)

3 函数赋给变量

var v\_obj = (funtion(a,b){

})(10,2)

4 闭包，函数内部的变量外部不能访问

对象易变性

继承

接口

注释描述接口：

/\*

interface CompoSite{

function add(child);

function remove(child);

function getchild(child);

}

interface FormItem{

function save();

}

\*/

继承

单体模式

var Singleton = {

attr\_name:'hzy',

attr\_age:19,

attr\_sucess:true,

method\_getname:function(){

},

method\_getage:function(){

}

}

访问：

Singleton.attr\_age = 25;

Singleton.method\_getname();