对象\*\*\*\*\*\*\*\*\*\*

## 一、Object的实例对象

### 1.对象的概念

#### 1）面向对象和面向过程

c语言面向过程的语言

Java js都是面向对象的语言

面向对象和面向过程是两个思想

自己干的，一步一步去做，对于自己来完成这个事情要面向过程

自己没干，一步一步都是别人干的，对于自己来说完成这件面向的是某一个对象

#### 2）什么是对象及作用

无序的名值对的集合（键值对的集合）就叫做对象；也是一种数据，对象类型数据

·如果存储一个简单的数据（一个数字，一个字符串） 直接var a = 10;

·如果存储一堆相同类型的数据 此时我们想到数组 数组就是专门用来存储多个数据用的

·如果我们想要执行一段代码，或者让这段代码有功能，此时我们需要函数

·如果我想描述一个复杂的事物，比如说一个人，一台电脑（需要用到多个属性或者方法才能描述清楚），此时就要用到对象；

。

#### 3）对象的创建方法

a)字面量创建:

b) new Object（构造函数定义）

无论你用的是上面的哪一种方式去创建的对象，这些对象都被称作是Object的实例对象

一般情况下，Object的实例对象，可以是任意的对象，你把它的属性写成是人的属性，那么这个对象就是一个人，你把它的属性写成是狗的属性

那么它就是个狗，所以我们说Object的实例对象是一个（万能的 通用的 可以表示任何事物的）；

### 2.对象的操作及遍历（增删改查）

增删改查：点语法和[]语法操作

遍历 for in循环进行遍历对象

数组是数组数组也是对象

函数是函数函数也是对象

对象就是对象，一切都是对象

**3.练习题**

  //1

var a = 'test'

    var test = 'test'

    var obj = {

      test:'测试内容1'

    }

    console.log(obj[test],obj.test,obj.a)

    // 2

    var bird = {

     size: "small"

    };

    var mouse = {

      name: "Mickey",

      small: true

    };

    // console.log(mouse.bird.size)

    console.log(mouse[bird.size])

    console.log(mouse[bird["size"]])

    // 3

    var a = { content: "世界!" };

    var b;

    b = a;

    b.content = "你好";

    console.log(a.content,b.content);

    // 4

    var a = {a:1,b:2};

    var b = a;

    b.a=3;

    b.b=4;

console.log(a.a, a.b, b.a, b.b);

## 二、构造函数创建特定实例对象

### 1.构造函数的基本概念，使用，作用

构造函数：本质上也是一个函数，只不过通常我们把构造函数的名字写成大驼峰；

在js当中，没有类的概念（5版本）,构造函数可以理解为类；

任何的函数都可以是普通函数，也可以是构造函数，就看你在怎么用；

### 2.对this的讲解

#### this介绍

this最终的本质上其实是一个对象，它代表着函数的执行者是谁；也可以说谁执行了这个函数，this就指向谁（代表是谁），不同场合this的指向不同

js当中，一般this有这些场合：

·如果this是在一个函数当中，this就指向window

·如果this是在一个方法当中，this就指向这个方法的对象

·如果this是在一个构造函数当中，this永远代表的是准备实例化出来的那个对象；

·如果this是在一个事件的处理（回调）函数当中，this代表的永远是发生事件的事件源（也就是发生事件的元素）；

·如果使用call和apply，this由自己指定；

#### window对象简介

浏览器窗口对象，代码执行的时候所有的一切都是包含在窗口对象下的.

#### 函数的使用方式

首先我们要清楚无论是什么函数，本质上都是函数，函数在使用的时候有多重用法。

·构造函数去用：如果当作构造函数去使用，需要在函数调用前面加上new才代表构造函数使用，构造函数在使用的时候，this指向是准备实例化出来的那个对象，并且构造函数如果没有return，它是可以返回实例化出来的那个对象的；当然也要看return后面的是什么；

·普通函数去用，也就是说没有加new，那么此时函数相当于没有做实例化过程，仅仅是给window对象添加了一些属性及属性值。如果没有return，返回的是undefined;

### 3.new关键字实例化对象的过程

1、开辟内存空间(堆)

2、this指向该内存（让函数内部的this）

3、执行函数代码

4、生成对象实例返回

### 4.原型对象和原型链 \*\*\*\*\*\*\*\*\*\*\*\*

#### 原型对象

什么是原型对象:

一个函数对象在定义的时候，伴随它出现的另外一个对象就是原型对象，原型对象默认是object的实例对象

显示原型对象和隐式原型对象概念

函数对象身上的prototype属性值（显式）

实例化对象身上的\_\_proto\_\_属性值（隐式）

这两个属性的值是同一个对象，就是我们说的原型对象

#### 原型对象的作用，

资源共享 节约内存

#### 原型链

描述的是对象在查找属性或者方法的过程

实例化对象在找属性的时候，先从自身去找看有没有这个属性，如果有，直接使用这个属性的值，如果没有，会继续顺着这个对象的隐式原型对象（\_\_proto\_\_）找到这个对象的原型对象（和它的构造函数的显式原型对象是同一个），看看原型对象是否存在这个属性，如果有就使用原型对象当中的这个属性值，如果还没有，再去找原型对象的隐式原型对象（默认就是Object显式原型对象），找到以后去看看有没有这个属性，如果有就使用这个属性值；如果没有就返回undefined(代表已经找到顶了)；

### 5. apply、call 、bind

任何函数对象都有apply和call方法

apply和call可以使用第一个参数传对象，让函数或者方法的执行者（this）指向这个对象；

函数或者方法.apply(对象，【函数的参数】)；

函数或者方法.call(对象，函数的参数1，函数的参数2)；

call和apply干了两件事：

1、调用的时候先把this指向改为p1

2、然后再去执行使用的方法

call和apply可以让一个对象执行另外一个对象的方法；

bind 也是去修改函数内部this指向，不过他会返回一个新函数，不会自动执行，需要咱们手动执行

### 6. instanceof

#### typeof 应用的场景

typeof 值 返回的是数据类型的小写字符串形式

数字 'number'

字符串 ‘string’

布尔 'boolean'

undefined 'undefined'

null 'object'

数组 'object'

函数 'function'

对象 'object'

typeof 可以判定5种：

数字 'number'

字符串 ‘string’

布尔 'boolean'

undefined 'undefined'

函数 'function'

#### instanceof 应用的场景

instanceof 用来判断一个对象是哪个构造函数的实例用的 A(对象数据) instanceof B（构造函数）

专门用来解决判定数组和对象的时候使用

#### ===

全等于可以用来判定null和undefined的时候使用；因为他们两个都是数据类型，但是这数据类型当中只有一个值；

a === null

a === undefined