# 对象

## Object的实例对象

### 1.对象的概念

#### 1）面向对象和面向过程

在js当中，可以说一切皆对象，js是一门面向对象的语言；

C就是面过程的语言

java python js都是面向对象的；

#### 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；

·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:

对象是由属性和属性值组成的，其实就是我们说的键值；

属性也可以分为属性和方法

var obj = {

name:'赵丽颖',

'age':32,

"sing":function(){

}

}

b) new Object（构造函数定义）

c) 工厂函数模式

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

## 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是在一个普通函数当中，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；

var obj = {

eat:function(){

this

}

}

#### 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.

**全局变量和全局函数会特殊对待，作为window对象的属性**

**局部变量它是没有这个特殊待遇的，既不是window的属性，也不是函数对象的属性，就是一句普通的代码；**

#### 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么对象，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关键字作用

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（把空间的地址返回）

请一定要画图；

### 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，原型对象是默认是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）

注意：只有函数对象身上才会有prototype这个属性

不是函数对象身上没有这个属性

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和call

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；

call和apply可以让一个对象执行另外一个对象的方法；

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指向改为你指定的第一个参数对象

2、然后再去自动执行使用的方法

### 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

通过typeof instanceof以及 === 可以让我们判定js当中所有的数据类型

## 值类型，引用数据类型，堆， 栈

### 1.值类型和引用数据类型的概念

值类型都有啥: 其实就是我们所说的基本数据类型

对象类型都有啥：引用数据类型就是我们所说的对象数据类型、复杂（复合）

### 2.堆和栈的概念

数据结构的两种，专门用来解决数据的存储和读取

内存：栈内存和堆内存（其它语言） js当中只有堆内存

栈内存比较小 但是块

堆内存比较大 但是慢

但是数据结构是分栈结构和堆结构的

栈： 栈的数据结构 FILO（有序） 桶

堆： 链表的数据结构 无序的可以随意添加和删除

### 3.堆和栈的图解

### 4.值类型和引用数据类型与堆栈的关系

堆空间的释放是靠垃圾回收机制进行的

当程序函数或者整个程序执行完成后，栈里面所有的东西都被释放销毁，堆当中的数据可能还在，只是没有任何的变量指向（引用），那么堆当中的数据就会变成垃圾对象。回收机制会在适当的时候将垃圾对象清理回收；

如果我们在程序当中需要去删除对象，那么就将这个对象的变量赋值为null，代表这个对象引用被改变，这个对象也就成了垃圾对象，其实删除对象就是让堆当中的对象数据成为垃圾对象；

var a = {};

a里面存了 {}地址 a指向了{}这个对象

a = null; 把a的指向断开了；

### 5.面试题讲解:

第一题:

var num1 = 10;

var num2 = num1;

num1 = 20;

console.log(num1);

console.log(num2);

第二题

var num = 50;

function f1(num) {

num = 60;

console.log(num);

}

f1(num);

console.log(num);

第三题

var num1 = 55;

var num2 = 66;

function f1(num, num1) {

num = 100;

num1 = 100;

num2 = 100;

console.log(num);

console.log(num1);

console.log(num2);

}

f1(num1, num2);

console.log(num1);

console.log(num2);

console.log(num)；

第四题

函数传参如果传的是基本数据类型和传引用（对象）有什么区别

var a = 10;

var b = 20;

function add(a,b){

a = 30;

return a + b;

}

add(a,b);

console.log(a);

function f1(arr){

for(var i = 0; i < arr.length; i++){

arr[i] += 2

}

console.log(arr);

}

var arr;

arr = [1,2];

f1(arr);

console.log(arr);

第五题

两个对象是同一个对象，不同的操作有什么不同

var a = [1,2];

var b = a;

a[0] = 20; 如果a = [20,2]；

console.log(b);

var a = [];//new Array()

var b = [];//new Array()

console.log(a == b);

第六题

function Person(name, age, salary) {

this.name = name;

this.age = age;

this.salary = salary;

}

function f1(pp) {

pp.name = "ls";

pp = new Person("aa", 18, 10);

}

var p = new Person("zs", 18, 1000);

console.log(p.name);

f1(p);

console.log(p.name);

## 内置对象JSON

### 1.什么是json,json作用

json是一种数据格式；现在我们大多数都是通过json数据格式进行前后端数据交互的，json本质上是一个字符串，简称json串

前端往后台传数据的时候，要传json格式的数据json串

在前端json串的格式原形就是对象或者对象的数组；所以我们要先把数据存储为对象或者对象的数组，然后转化为json串进行传递

### 2. JSON对象的方法: parse， stringify

## Math工具对象

### 1.Math对象方式的使用

round, floor,ceil, random, max min PI pow abs sin

round, floor,ceil 都是要把小数转化为整数，如果是整数了，不动；

### 2.案例：

随机数点名

随机生成验证码

## Date日期对象

### 1.Date对象的方法

var date = new Date();

console.log(date);

console.log(date.getFullYear());

console.log(date.getMonth());

console.log(date.getDate());

console.log(date.getHours());

console.log(date.getMinutes());

console.log(date.getSeconds());

console.log(date.getYear());//已经废弃

console.log(date.toLocaleTimeString());

console.log(date.toLocaleDateString());

console.log(date.getTime());//1970 年 1 月 1 日之间的毫秒数

### 2.案例：

1. 格式化日期 2019年8月24日 下午4：43：

2. 封装函数实现格式化日期

## 包装对象

基本数据类型也可以使用.调用方法

作业：

对象的增删改查，构造函数

1、创建两个同一品牌的手机对象（要求品牌和方法在原型当中定义共享使用）;接下来把实例化对象的过程以及原型的原理图 每人画一遍；

2、封装自己的工具函数实现随机验证码功能，要求验证码的位数可以根据参数变化

3、函数传参如果传的是基本数据类型和传引用（对象）有什么区别；

4、把当天封装的随机数以及日期函数必须搞定

5、把值类型和引用类型面试题做一下，记得画图；