**综述**

**一、思路:**

**面试题需要时刻准备**

[开除员工视频](https://www.bilibili.com/video/av81757640?from=search&seid=9788166156687721051)

所以时刻准备着面试题

为什么我要放个视频, 因为后面可能有一瞬间比较枯燥

**看下这几道题, 懵了吗?**

js基本类型 ?

是否成立?

[] == ![]

const b = new Object(11)

console.log(b)

const obj = {

toString() {

return 'abc'

},

valueOf() {

return {}

},

}

const res = obj + 2

console.log(res,'res')

定义一个a, 让下文成立

if (a === 1 && a === 2 && a === 3) {

console.log("Why hello there!")

}

定义一个a, 让下文成立

if (a == 1 && a == 2 && a == 3) {

console.log("Why hello there!")

}

**讲解基础知识**

只要知道概念, 互相转换, 基础的题目会做

**解题**

**js数据类型**

**一 js数据类型分类:**

标准的回答是: 7种

6种原始类型 和 引用类型 // 引用类型的细分不算作js数据类型的区分

**原始类型/原始值:**

string,number, boolean, null, undefined, symbol(ES6新增，表示独一无二的值)

tip: NaN属于number类型

是存放在栈(stack)内存中的简单数据段，可以直接访问，数据大小确定，内存空间大小可以分配

**引用类型/对象值**

function,object,array等可以可以使用new创建的数据

存放在堆(heap)内存中的数据，如var a = {}，变量a实际保存的是一个指针，这个指针指向对内存中的数据 {}

**valueOf & toString**

**一 概念:**

在Object.prototype上定义了valueOf, toString两个方法

这两个方法在类型转换时会被用到

这两个方法的返回值有默认的规则

当然也可以在Object的子类上重写这两个方法

**二 valueOf, toString的默认值:**

就是Object.prototype上定义了valueOf, toString两个方法的返回值

**1 toString**



**2 valueOf**

除了Date类型的对象, 其他的都是返回自身



**三 对象的原始值:**

对象的原始值可以从valueOf或者toString的返回值选其一

根据顺序有两种选对象原始值的方法

**valueOf -> toString**

if obj.toString() 是原始值, 则将该值作为对象的原始值

else if obj.valueOf() 是原始值, 则将该值作为对象的原始值

否则报错

**valueOf -> toString**

if obj.valueOf()是原始值, 则将该值作为对象的原始值

else if obj.toString() 是原始值, 则将该值作为对象的原始值

否则报错

**数据类型转换**

**一 概念:**

原始值到原始值: 就是计算 String(a) 或Number(a) ,

原始值到对象值 : 计算new Object(a).... 的值

对象值到原始值: 就是计算 String(a) 或Number(a)

[参考](https://juejin.im/post/5b6906b46fb9a04fcb5b8771)

**二 原始值到原始值:**

**1 原始值转化为布尔值**

所有的假值(undefined、null、0、-0、NaN、””)会被转化为 false，其他都会被转为 true

**2 原始值转化为字符串**

都相当于原始值外面包一个双引号 ""

**3 原始值转为数字**

**布尔转数字**：true -> 1, false -> 0

**字符串转数字：**

字符串是纯数字内容, 或只有首尾有空格则直接将数字部分作为返回值

空字符串返回 0

否则返回NaN

**其他类型转数字**

null -> 0 , undefined -> NaN

**三 原始值到对象:**

根据原始值的类型调用对应的构造函数, 返回实例

null, undefined 转换后是: {} // 空的Object实例

const a = new Object('aa')

console.log(a) // String {"aa"}

const b = new Object(11)

console.log(b) // Number {11}

**四 对象到原始值:**

**1 对象转为布尔**

都为 true

**2 对象到字符串, 数字**

**对象到字符串**

按照toString -> valueOf的顺序获取对象的原始值

将原始值转为字符

**对象到数字**

按照valueOf -> toString的顺序获取对象的原始值

将原始值转为数字

**例子**

var obj = {

toString() {

return {}

},

valueOf() {

return 123

},

}

var res = String(obj)

console.log(res,'res') // '123'

尝试调用toString方法, 但是返回值不是原始值, 所以调用valueOf, 并将123转成 ‘123’返回

**运算符和的隐式转换**

**一 概念:**

在进行运算符计算的时候, 当参与计算的值类型不同时 ,为了让计算逻辑正确, 有可能会将参与计算的值转换之后再计算

tip:

这里的值的类型是指: js数据类型

**二 == , ===运算符:**

**1 ===**

1 不带任何类型转换, 原始类型必须完全相等, 引用类型地址必须相等, 才返回 true

2 NaN 不等于 NaN

**2 ==**

1 绝大多数据比较, 将两个值都转换成数字类型, 然后比较

tip: 只有在值的类型不同的时候才转换, 相同则不转换

2 null和undefined之间相等（==），并且也与其自身相等，但和其他所有的值都不相等

3 NaN 和任何数据不相等

**三+ 运算符:**

**1 作为一元运算符**

+a 相当于 Number(a)

**2 作为二元运算符**

依次按照下面的顺序操作:

1 将对象按照valueOf -> toString的顺序变成对象原始值

tip: 日期对象会按照 toString -> valueOf的顺序

2 如果有字符串, 将两边都转成字符串再拼接

3 其余情况将两边都转成数字, 相加

**例子:**

**例子1**

const res = null + undefined

console.log(res,'res') // NaN

将两边都转成数字, null -> 0, undefined -> NaN , 相加是NaN

**例子2**

var obj = {

toString() {

return 'abc'

},

valueOf() {

return {}

},

}

var res = obj + 2

console.log(res,'res') // 'abc2'

将obj转成原始值 -> ‘abc’,然后将 2 转成字符串 ‘2’, 然后做拼接

**题目解析**

**一 不可能的逻辑判断1:**

定义一个a, 让下文成立

if (a == 1 && a == 2 && a == 3) {

console.log("Why hello there!")

}

[参考](https://my.oschina.net/zhangyafei/blog/1941407)

**解法1 valueOf**

let i = 1

var a = {

valueOf() {

return i++;

},

}

利用类型转换会执行valueOf方法的性质, 结合一个计数变量

**改写成闭包:**

const a = {

valueOf: (function () {

let i = 1;

return function () {

return i++;

}

})()

}

if (a == 1 && a == 2 && a == 3) {

console.log('22');

}

**解法2 数组toString**

var a = [1, 2, 3];

a.join = a.shift;

if (a == 1 && a == 2 && a == 3) {

console.log('22');

}

==运算符隐式转换, 将a转化为数字, 进行比较

就是按照valueOf -> toString的顺序取原始值, 然后将该值转化为数字

valueOf没有返回原始值, 所以用toString的值

arr.toString返回arr.join(), 所以只要将arr.join强行重写成arr.shift就行了, 正好每比较一次就只想执行一次, 删除并返回数组第一个元素,

**二 不可能的逻辑判断2:**

定义一个a, 让下文成立

if (a === 1 && a === 2 && a === 3) {

console.log("Why hello there!")

}

[参考](https://my.oschina.net/zhangyafei/blog/1941407)

**解法1 劫持window的属性**

数据劫持, 利用所有全局变量都是window属性的特性

let val = 1;

Object.defineProperty(window, 'a', {

get: function () {

return val++;

}

});

**伪解法2 隐形字符**

var aﾠ = 1;

var a = 2;

var ﾠa = 3;

if (aﾠ === 1 && a === 2 && ﾠa === 3) {

console.log("Why hello there!")

}

注意if里面a后面, 前面的空格，它是一个Unicode空格字符，不被ECMA脚本解释为空格字符(这意味着它是标识符的有效字符, 其实定义了三个变量)。所以它可以解释为

var a\_ = 1;

var a = 2;

var \_a = 3;

if (a\_ === 1 && a === 2 && \_a === 3) {

console.log("Why hello there!")

}

tip: 这个空格是打不出来的 !

**三 求值: [] == ![]:**

**解答**

1 先运算![] , 结果是false

[] == false

2 将== 左右两边转成数字,

右边false -> 0

左边 [] -> []的原始值是空字符串, 空字符串转数字 -> 0

3 答案: true

**话题2**

**一、总览:**

**1. 概念**

**一、话题:**

**1. 概念**

**细项1**