一．变量分为原始值和引用值(由值决定类型)

1> 原始值 存在stack(栈)里面

(1)Number(数字类型)

(2)String(字符类型)

(3)Boolean(true/false)

(4)undefined(没有定义)

(5)null(无意义空 占个位置)

2>引用值 大部分放在heap(堆)里面

(1)array(数组)

(2)Object(对象)

(3)function(方法)

(4).......date regExp

1. 布尔值为false的情况(除了这些boolean都为true)
2. undefined（未定义，找不到值时出现）
3. null（代表空值）
4. false（布尔值的false，字符串"false"布尔值为true）
5. 0（数字0，字符串"0"布尔值为true）
6. NaN（无法计算结果时出现，表示"非数值"；但是typeof NaN==="number"）
7. ""（双引号）或''（单引号） （空字符串，中间有空格时也是true）

这样测试 -->

打印出false 或者 使用Boolean(a)测试

三．与( || )或( && )非( ! )问题

1> 或( && )从第一个往后依次判断是否位假, 是假则返回其值, 若全为真则返回最后一个值 -->

打印出的结果为3

1. 与( || )从第一个往后依次判断是否位真, 是真则返回其值, 若全为假则返回最后一个值 -->

打印出的结果为1

3> 非( ! )表达式转化成布尔值取反

注意: !!就是取反再取反, 没有发生改变

四．typeof详细讲解(检验值类型)

1>typeof()返回值的六种类型

(1)number

(2)string

(3)boolean

(4)object(null数组对象都是返回这个)

(5)undefind

(6)function

2>typeof的显示类型转换(注意其大小写)

(1)Number(mix)返回值为mix, 返回类型为number类型

注意：若为字符返回NaN, 且mix有小数则报错

(2)parseInt(a, radix)返回值为a成radix为底的十进制数, 返回类型为数字类型

注意：若numb不是radix进制格式的数则返回值为NaN

(3)parsFloat(str)返回值为str(有小数点小数点原样返回), 返回类型为number类型

(4)String(str)返回值为str(原样返回), 返回类型为string类型

(5)str.toString(radix)返回值为str以radix的数, 返回类型为string类型

(6)Boolean(str)返回值为str的true/false, 返回类型为boolean

测试方法 -->

(1)值打印出65,

(2)类型打印出number

3>隐式类型转化

(1)isNaN(a)a转化成number与NaN进行对比,相同true,不同false 返回类型为boolean

(2)++/-- +/-(一元正负)转化成Number在相加, 返回其值, 返回类型类为number

(3)+ 转化成string在相加, 返回其值 返回类型为string

注意：两边有一个是字符串才转化

(4)-\*/% 转化成Number在相乘, 相减.. 返回类型为number

注意：若有字符串则会返回NaN

(5)&&||！数字优先原则(全为数字返回数字), 返回值为true/false 返回类型为boolean

(6)<><=>= 转化成数字优先原则, 返回值为true/false 返回类型为boolean

(7)== !== 转化数字优先原则, 值为true或者false 返回类型为boolean

注意：undefind和null不大于等于小于零且undefind=null是true,NaN=NaN是false

===和！==不发生隐示类型转换

1. 定义函数
2. 函数声明

(1)function () {}(匿名函数, 一般立即执行函数这样声明)

(2)function test() {} 执行函数直接test()

注意：函数声明开发规范写明命名函数需是小头封式(theFirstName)

1. 函数表达(使用一个变量来接收这个函数)

(1)var test = function test () {} (命名函数表达式) test无意义

(2)var test = function abc () {} (匿名函数表达式) 常用

注意：两者唯一的区别就是查看test.name属性时写了属性数abc没写则是test

1. 实参和形参

1>概念

function sun(a, b) { c = a + b console.log(c) } sun(1, 2)

a, b是形参

1, 2是实参

2>实参长度和形参长度

(1)形参长度 --> arguments.length

(2)实参长度 -->函数名.length

3>映射关系(形参和实参对应时才有) -->

function sum(a, b) { b = 2; console.log(arguments[1]);} sum(1);

打印出undefined, 打印不出2

sum(1)改为sum(1, 随便什么参数) 打印出2

1. return作用

1>函数有return时结束 函数没写return在函数的最后系统会隐示的自动加上return.

2>.能返回值。(用这个时候一般用一个变量接收这个值)

1. switch case

switch case用法的基本形式 -->

switch (表达式) case 条件1 执行函数 case 条件2 执行函数 ...

注意：(1)它是打印出符合条件之下的所有语句(穿透)。

(2)break可终止穿透, break只能放在循环里面, 不然报错。

(3)continue是继续 终止本次循环执行下一圈循环

1. while循环的用法(注意和if循环区分)

while循环用于不确定循环次数, if循环可以控制次数(满足条件无限循环里面的东西,

知道不满足条件) while(条件){循环的东西}

1. Js执行的三部曲(预编译)
2. 语法解析(通篇扫描下有没有低级的语法错误 比如少了分号没有)
3. 预编译

(1)预解析的详情讲解过程(函数整体提升, 变量声明提升)

##创建AO对象(Active Object)

##找形参和变量声名，将变量和形参名作为AO属性名，值为undefind

##将实参和形参相统一(实参值赋值给形参)

##在函数体里面找函数声明，赋值予函数体

1. 解释执行(解析一行执行一行)
2. 作用域及和其相关的东西(里面的可以访问外面的, 外面的不能访问里面的)
3. 变量的作用域类型

(1)全局作用域(window, GO)

##未经声明的变量赋值, 此变量全局对象所有(window), 称暗示全局变量

##全局上的任何变量都是window所有

(2)局部作用域(AO)

2>[[scope]]属性, 及作用域

(1)属性其里面存储了执行期上下文(AO, GO)

(2)scope就是javascript函数(对象)中一个我们不能访问的属性, 仅供javascript引擎使用的属性

3>作用域链

[[scope]]中所存储的执行期上下文对象的集合，这个集合呈链式链接，我们把这种链式链接叫做作用域链。

4>执行期上下文

当函数执行时，会创建一个称为执行期上下文的内部对象。一个执行期上下文定义了一个函数执行时的环境，函数每次执行时对应的执行上下文都是独一无二的，所以多次调用一个函数会导致创建多个执行上下文，当函数执行完毕，执行上下文被销毁。

5>闭包

闭包就是里面的函数被保释到了外部(在销毁之前保存到了外部)

1. 立即执行函数(与普通函数唯一的区别是执行后马上释放)
2. 写法

(1)( function (形参) {} (实参) )

(2)( function (形参) {}) (实参)

注意：function () {}是匿名函数

1. 实用用法

(1)return出一个值, 使用一个变量接收

注意：(1)只有表达式能被执行符号执行, 且被执行符号执行的函数会被永久放弃 -->

##var test = function () { console.log('a'); } () 输出a

##function test() { console.log('a');}() 输出不了a, test输出不了函数

(2)里面是表达式 (function f() {})里面的函数变成了函数表达

1. 对象
2. 构建对象的方式(常用, 解决了下面所毛病)

(1)自定义构造函数创建对象

##function 大头峰式命名 () {} var 变量(小写) = new 前面声明的函数() -->



##new对象时候详细讲解



(2)调用系统构造函数创建对象(无法识别是属于具体的哪个对象)

##var 变量名 = new Object() //用个变量接收叫实例化对象 -->



(3)工厂模式创建对象(能批量生产对象,且能识别类型，但繁琐) -->

##function 变量() { var obj = new Object(); 添加属性方法在此添加 return obj;}

var 变量接收 = 变量() (函数执行); (接收变量和变量可以不同) -->



注意:return obj一定要写不然打印出的对象是undefined

(4)通过字面量对象创建对象var obj = {}

(5)Object.create(proto[,propertiesObject])(proto: 对要继承的原型)(propertiesObject: 对象，可选参数，为新创建的对象指定属性对象。)

2>循环对象的方法

(1)for prop(随便取, 其值为对象的key) in 对象

3>对象的增删改查

(1)增 对象.东西(里面没有的东西) = "值" -->

mrZhu.age = "21" 则在1的例子里面加上了 age : 21,属性。

(2)查 对象.东西(里面有的东西) -->

mrZhu.sex 里面就会显示male

(3)改 对象.东西(里面有的东西) = "值" -->

mrZhu.smoke = "1年" 对象里面smock就会显示1年。

(4)删 delete 对象.属性(里面有的属性) -->

delete mrZhu.sex 对象里面就没有sex属性了。

4>构造函数的内部原理

(1)在函数题前面隐示的加上this = { }

(2)执行this.xxx = xxx

(3)隐示的返回this -->

function Person(name, height) {

//var this = {}

this.name = name;

this.height = height;

this.say = function () {

console.log(this.say)

}

//return this;

}

注意：只有new Person()之后才发生构造函数的三部//，否则就是正常函数的执行。

5>对象小知识总结

(1)当一个变量没经过声名就使用会报错，但当一个属性没有使用会输出undefined

(2)调用属性的两种方式

##obj.属性 (使用obj.属性里面会隐示的转化下obj[name])

##obj['属性'] 应用(实现调用方法数字几就出几的名字)-->

var Deng = {

wife1 :{name : "xiaozhang"},

wife2 : {name : "xiaozhang"},

saywife : function (num) {

return this["wife" + num];

}

}

1. 原型
2. 原型的基本概念

(1)型是function对象的一个属性，它定义了构造函数制造出了对象的共有祖先。

(2)原型即prototype，函数刚出生原型就定义好了。它类似一个空对象。

(3)函数会继承原型上的属性和方法(可以利用其, 提取出共有属性)。

注意：若自己身上有原型上一模一样的属性时出现自己身上的属性。

2>原型的增删改查(四个差不多, 以增为例子)

Car.prototype.height = 1400;

function Car(color) { this.color = color } var car = new Car(）；

car.prototype.height = 1000;

注意：添加需要.prototype

3>constructor(构造器), prototype和 \_\_proto\_\_

(1)constructor(构造器)

(2)prototype

(3)\_\_proto\_\_

(4)其他

##\_\_proto\_\_和constructor是对象独有的属性(在js中对象是由函数构成的所以有这两个属性)

##prototype是函数独有的属性(函数也是对象的一种，是并集关系不是父子集关系)

##hasOwnProperty(属性的字符串形式), 判断是不是原型上的属性

##in 使用形式 -->"属性名" in obj

注意：in判断原型上的属性和hasOwnProperty一样, 但不同的是它会把自己的原型也算上来

4>原型小知识总结

(1)object.prototype是所有原型链的终端

(2)原型链不能增删改查, 只能自己修改自己原型

(3)Object.creat( 对象或者null )

##对象的话原型就指向这个对象

##null对象不最终继承Object.prototype除了这个全是最终继承Object.prototype

(4)人为设的\_\_proto\_\_不会继承(写了Object.creat(null)截断了继承对象，然后人为设个\_\_proto\_\_指向是不会继承的)

(5)constructor是Object()就是原型链的顶端

(6)因为原始值undefind和null不是对象，也没有原型链所以不可以经过包装类所以不可以添加属性和方法

十五．包装类

1>包装类基本概念

(1)普通变量不能直接调用属性和方法, 而对象可以直接调用属性和方法

这些普通类型的变量会默认会调用包装

包装类: 普通类型的变量经过包装可以直接调用属性和方法 -->



(2)基本包装类型：数字(number), 字符串(string), 布尔(boolean)

2>包装类小知识总结

(1)原始值不能有属性和方法的为什么能调用？

经过包装类(不能加属性但又能访问属性)

1. 常见的字符方法(都是静态方法)及字符串小知识

1>.charAt(index) 返回对应字符串的第index位字符 -->

 返回值为h

注意: 超过字符串长度(length-1)返回空字符串,空格也占一个字符索引位。

2>String.fromCharCode(ussc码, ussc码, ...)返回参数对应的USSC码对应的字符串

 返回值为ABC

3>.indexOf(要找的字符串, [从某个位置开始找(索引值)])

找参数字符串的索引位(没找到返回-1) -->

注意区分：lastIndexOf()从后先前找，获取的索引仍然是从左往右数

 返回值为0

3>.replace(“old string”, “nwe string”)方法 把old string替换成nwe string

 返回值为olllll

注意：不用正则只替换一次

4>.slice(“从第几位”, “到第几位”)返回提取后的字符串(第几位都是索引)

 返回值为h

5>.split(“删除什么字符串”, [“切割后留下几个”] )按什么字符串拆分 返回多个数组

 返回值为[“a”, “b”]

6>.substr(“开始的位置”, [“个数”])返回截取后的新字符串

 返回值为el

7>.substring(“开始的索引”, “结束的索引”), 返回截取后的字符串，不包含结束的索引字符串

 返回值为he

8>.toLocaleLowerCase(); 把字符串转为小写的 和toLocalecase();一模一样

 返回值为hello world

9>.toUpperCase(); 把字符转为大写

 返回值为HELLO WORLD

10>.trim(); 去除字符串两端的空格



返回值为

1. 字符串小知识

(1)字符串string和字符串对象(new String())

-->

 输出还是hello 无法改变(只可以读，不可写)

-->

栈指向了新的堆

输出的结果是”小苏好邪恶”

1. 数组
2. 构建数组的方式

(1)var arr = [];

(2)var arr = new Array();

注意：这个括号和(1)构建的对象[]添加值一样唯一的区别就是只传一个值这个代表数组长度。

1. 数组小知识

(1)当给数组第十位赋值时会将数组撑长为11 -->

arr[10] = abc 打印出 [undefined\*10, abc], 且length值为11

(2)数组除了可以用for循环遍历外还可以使用for item in arr(数组是特殊的对象)

3>数组方法

(1)可以改变原数组的方法

##.push(); 把值追加贷数组中(最后面), 返回值为追加后数组的长度 -->



数组为 打印arr为 5

##.pop()把删除数组中最后一个元素的值, 返回值为删除的这个值



数组为 打印arr为400

##.shift()删除数组中第一个元素, 返回值为删除的这个值



数组为 打印arr为 100

##.unshift(值)向数组中的第一个元素前插入值, 返回的是插入后数组的长度



数组为 打印arr为5

##.sort(function () {})排序不稳定, 可传入一个比较函数可以进行排序

function () { 排序规则 } 升序原理(return a-b;)降序(return b-a)

注意：1形参一定要写两个值

2看返回值

1)当返回值为负数,那么前面的数放在前面 (return 1和return -1)

2)当为正数,那么后面的数在前面

3)为0,不动

##.reverse()数组反向



数组为 打印arr为[25, 16, 4, 1]

##.splice(“开始”, “要删除的个数”, [“截取后添加的元素”])截取切在截取位置添加元素

注意：截0位(在数组中间添加元素)



数组为 打印arr为4

(2)不改变原数组(一般用个函数接收不然没意义)

##.concat()连接两个数组



数组为 打印arr为原数组arr

##.join("东西"按东西样式连接(注意需要时字符串形式))



数组为 打印arr为原数组arr

##.split("东西"按照东西样式拆分数组)



数组为 打印arr为

##.slice(“开始”, “结束“);截取数组从开始到结束(不包括结束那个元素)



数组为 打印arr为原数组arr

4>其他数组方法

(1)A instanceof B(为Array或者Object) 检测A是否属于B -->





注意：数组是特殊的对象上例子中写 [] instanceof Object 返回值为true

(2)Array.isArray(数组) 判断是否是数组 -->





(3)every(function () {}) 可用于数组判断

##数组中的每个元素都会执行这个函数

##函数的第一个参数代表元素的值

##第二个代表元素的索引

##第三个是原数组(谁调用的就是谁一般用不到第三个参数)

##返回的是布尔值, 所有都满足条件才返回true -->





(4)filter(function () {})筛选

##返回符合设定条件的值成数组(新数组)形式 数组中的每个元素都会执行这个函数##函数的第一个参数为元素的值

##第二个元素的索引

##第三个参数为遍历的数组(原数组) -->





(5)forEach(function () {}) 循环数组(相当于一个for循环)

##第一个参数为元素的值

##第二个为元素的索引

##第三个参数为原数组

(6)indexOf(元素值) 返回的是元素的索引, 没有这个元素值则返回-1 -->





(7)map(function () {}) 数组中每个元素都会执行这个函数，执行后的结构重新放到一个新数组中返回出来





1. this问题

1>this指向问题

(1)函数的预编译过程this执向window

new test();new完之后会预编译第一时候this = Object.creat(test.prototype)

(2)全局作用域this指向window

(3)call/aplay可以改变this指向。

(4)obj.fun(); fun()里面的this指向obj(谁调用这个方法this指向谁) -->

var obj = { a : function () { console.log(this.name) }, name : "abc" }

obj.a(); 这个this指向obj

2>this小知识总结

(1)函数的连续调用return this（函数不屑return默认的值是undefined）

1. 区分数组[]和对象{}的几种方法

1>.obj.constructor打印function Object(){[native code]}

2>.[] instanceof Array 打印ture

3>Object.prototype.toString.call([]); 打印出'[object array]'

注意：var obj = {} (直接在控制台上打印{}不可以会报错)

1. Data对象

1>不需要先new Date的方法

(1).new Date() 获取当前时间 -->





(2).new Date(string) 传入字符串参数的时间 8:00 -->





注意:需要是字符串，获取的是时间类型的值

(4)计算时间 -->





注意：是数字类型 为了计算，时间值和时间值不可计算可以用其计算时间



2>需先new Date的方法(常用)

(1).getFullYear()获取年的方法用 -->





注意区分：.getYear()方法不常用(1900到现在的年数)

(2).getMonth() 获取月份方法 -->





注意：获取的月份从0开始要 +1才是当前的月份 例子中真实是7月

(3).getDay- 获取星期几 -->





(4).getDate() 获取日期 -->





(5).获取小时,分钟，秒



(6)其他方法

##日期









##小时分钟秒









##毫秒数





注意：和Date.now()区别

-- Date.now和new Date是同级， 而value需先new Date在使用

二十. 内置对象Math(大部分用来处理数字) --> 查看js文档百度搜索MDN--》

<https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Math>

注意：Math不是一个构造函数不像Object -->

不能使用new Math实例化一个方法(报错)

调用里面的方法Math.方法，new Object是使用实例化变量.方法来调用

Math.PI();(PI方法在里面已经定义好了) obj.see();（需在里面定义了）

封装一个Math.max()方法-->



二十一. 静态方法(Math.radaio())和实例方法(new Date().方法)

