变量：存储数据或者操作数据

原始数据类型：number, string, boolean, null, undefined, object

Number:数字类型（整数和小数）

不要用小数验证小数，有BUG，历史遗留问题双精度不精确

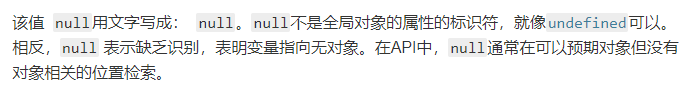
String:字符串类型（值一般用单引号或者双引号括起来）

Boolean：布尔类型（ture（1）和false（0））

Null：空类型，值只有一个：null，想要一个对象指向为空，此时可以赋值为null

变量的值如果想为null，必须手动赋值





Undefined：未定义，值只有一个：undefined

变量声明了，没有赋值，结果就是undefined

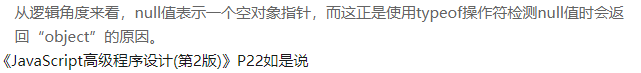
函数没有明确的返回值，如果调用了，结果也是undefined

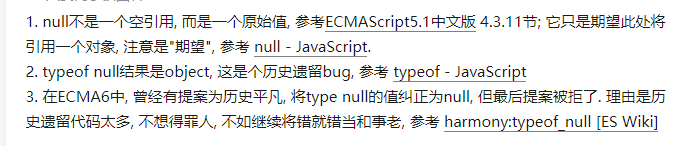
Undefined与数字相加和Number类型转换结果是：NaN

Obeject：对象

判断变量的数据类型：typeof（变量名）只能判断6个基本类型（原始类型）

用typeof判断null的数据类型结果为：object ==>





<https://juejin.im/post/5a439e30f265da4335630bf4>

NaN:不是数字

isNaN判断变量是不是数字，数字为false，不是数字为true

转义符：右斜杠\

字符串拼接：使用+可以把多个字符串放在一起形成一个字符串

只要一个是字符串的，其他是数字的相加也是拼接效果

一个字符串和一个数字相减会将字符串转变成数字，然后相减

其他类型转数字类型

parseInt（）；转整数，从左往右判断直到遇到不是数字的停止转换，开头就不是数字的输出结果为NaN

ParseFloot（）；转小数

Number（）；转数字，比较严格，有个不是数字的都是NaN

Number（null）：0

Number（“ ”）: 0

Number(“sdgdf”):NaN

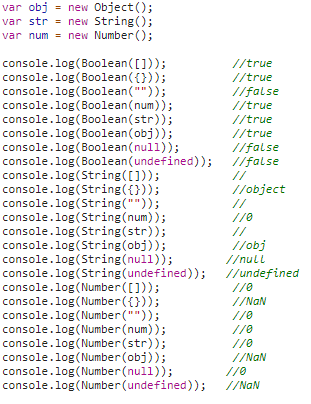
其他类型转字符串类型

toString（）变量有意义则使用tostring（）使用没有意义的变量(null,undefined)转换则会报错

String（）变量没有意义则使用String（）

String(null):null

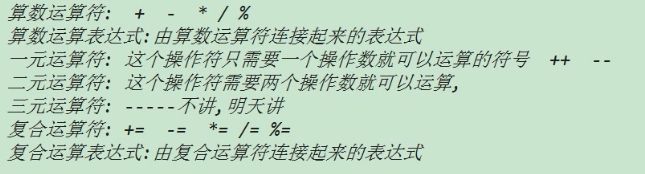
String(undefined):undefined

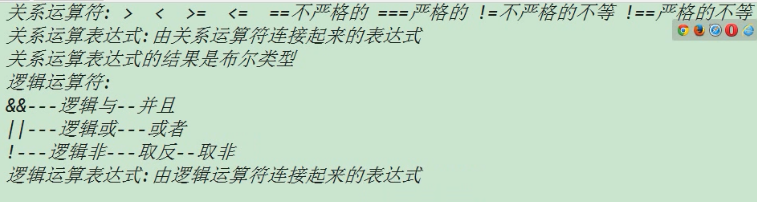
疑问？？？

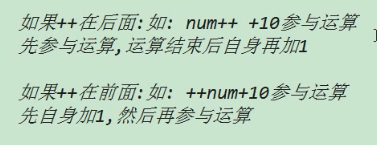
其他类型转换布尔类型



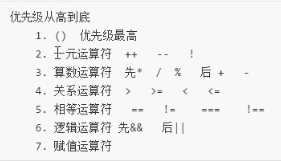
操作符







Switch-case中的判断是严格模式的比较（值和类型）



字面量：把一个值赋值给一个变量

Var n = 1; 字面量

Var n = y：不属于字面量,因为两个都是变量

数组：一组数据，有序的数据，一次性存储多个数据

通过构造函数创建数组

Var 数组名 = new Array();

通过字面量方式创建

Var 数组名 = [];

如果数组中没有数据，但是有长度，数组中的每一个值都是undefined

数组存储数据的类型可以不一样

函数：将一堆重复的代码封装，需要的时候再调用

函数名字：遵循驼峰命名法

重名函数会覆盖

函数返回值：在函数内部有return关键字，并且关键字后面有内容，

如果一个函数没有明确的返回值，那么调用的时候接收了，结果就是undefined

（没有明确的返回值：函数中没有return，函数中有return，但是return后面没有任何 内容）

Return 语句下面的代不会执行

arguments伪数组对象：函数在调用的时候，传入的几个参数

Arguments.length

使用arguments对象可以获取传入的每个参数的值

函数表达式： 把一个函数赋值给一个变量，此时形成了函数表达式

Var 变量 = 匿名函数;

Var fn = function (){

};

如果是函数表达式，那么此时前面的变量中储存的就是一个函数，而这个变量就相当于一个函数，可以直接加小括号调用

函数表达式后面，赋值结束后必须添加分号

函数的自执行，没有名字，声明的同时直接调用

(function(){

})();

函数的数据类型为：function ( typeof fn)

函数可以作为参数使用，如果一个函数作为参数使用我们将这个函数称之为回调函数

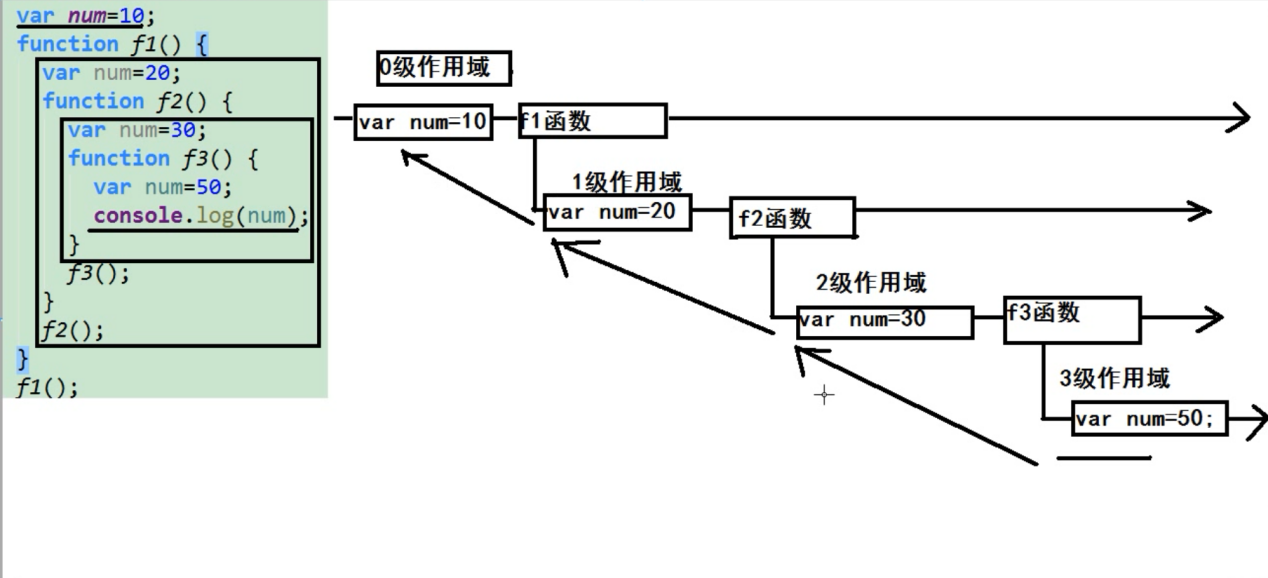
函数可以作为返回值使用

作用域：使用范围

隐式全局变量：声明的变量没有var，就叫隐式全局变量

定义变量使用var是不会被删除的，没有var声明的隐式全局变量可以被删除

作用域链

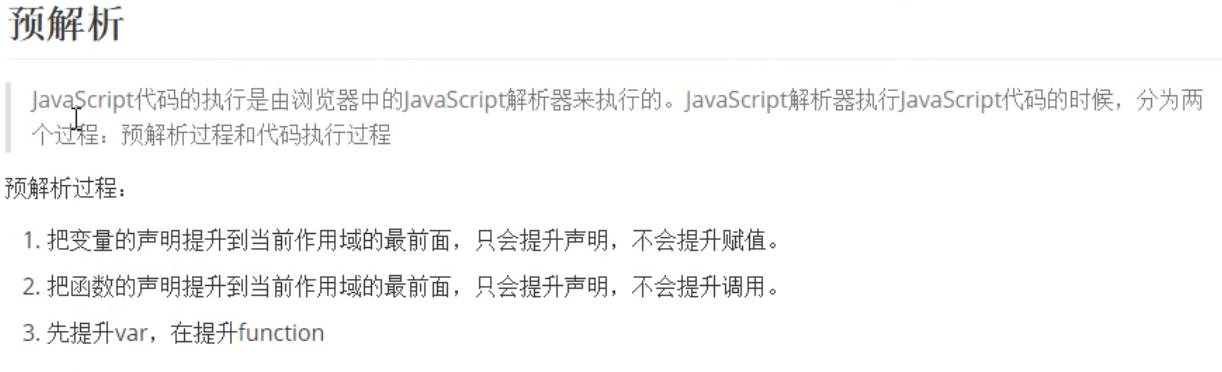


预解析：提前解析代码

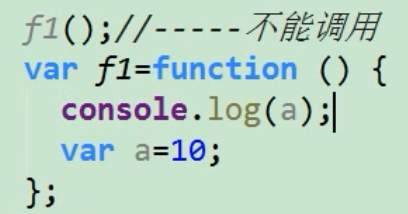
声明提前：变量声明，函数声明

变量声明提前在最上面，才到函数声明，如果函数名和变量名重叠，则函数名会覆盖掉变量名

提前到当前所在作用域的最上面



函数表达式未赋值之前调用会报错



对象：有属性和方法，特指的某个事物

一组无序属性的集合，属性的值可以是任意类型

面向对象：封装,继承,多态

Js不是面向对象的语言,但可以模拟面向对象的思想

Js是一门基于对象的语言

创建对象的三种方式

调用系统构造函数创建

Var 对象名 = new Object();

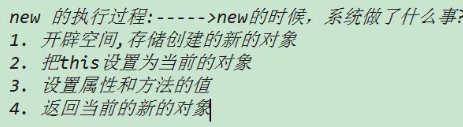
调用自定义构造函数创建 ===>使用new创建对象每次都会在堆中开拓新的空间

字面量方式创建

缺陷：一次性对象，改属性和方法需要在源代码上改

实例化对象

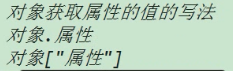
Var obj = new Object();

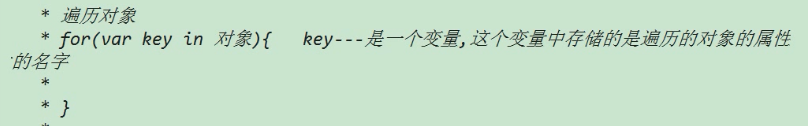


对象特征：属性,方法,行为

添加属性：对象名.属性名 = 值

添加方法：对象名.方法名 = 函数

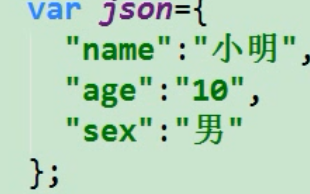




自定义构造函数与函数的区别是函数名的拼写,自定义构造函数的名字首字母为大写

JOSN格式的数据：一般都是成对的，是键值对

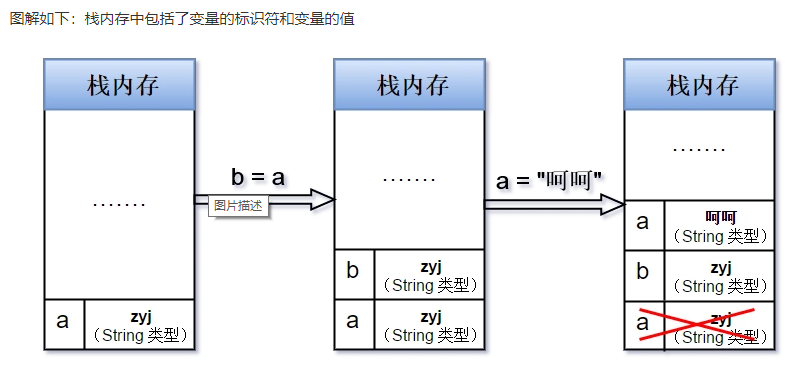
Json也是一个对象，数据是成对的，一般json格式的数据无论是键还是值都是用双引号括起来



数据的简单类型和复杂类型

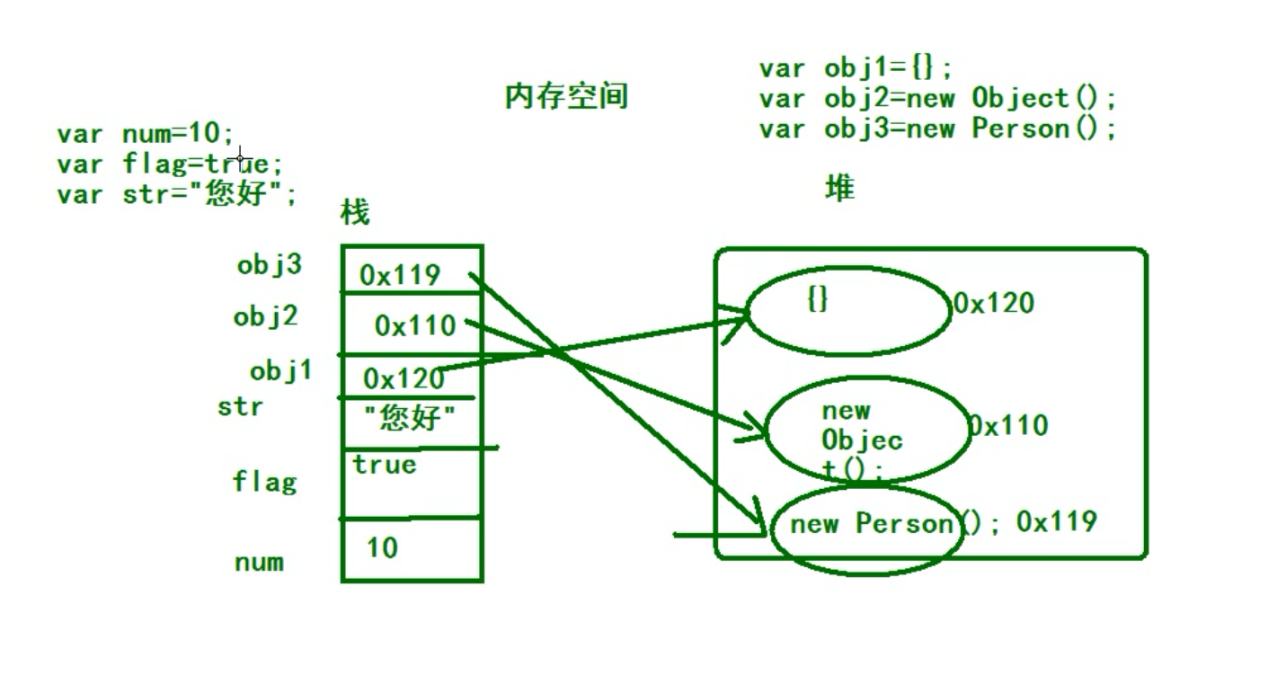
基本类型（简单类型）,值类型：number,string,boolean ===>栈中存储，值类型中的传递，传递的是值（复制）





复杂类型（引用类型）:object ===>地址在栈中存储和对象在堆中存储，引用类型之间的传递，传递的是地址（引用）

空类型：undefined,null



Js中的三种对象

自定义对象：自己自定义的构造函数创建的对象

浏览器对象：BOM

内置对象：js系统自带的对象

Math

Math.ceil():向上取整

Math.floor():向下取整

Math.abs():绝对值

Math.random():随机0-1之间的数

Math.max():一组数中的最大值

Math.min():一组数中的最小值

Math.Pi:圆周率的值

Date

Var Dt = new Date(); ==>创建时间对象，返回当前的时间

Dt.getFullYear():年份

Dt.getMonth()+1:月份，从0开始所以要加1

Dt.getDate():日期

Dt.getHours():小时

Dt.getMinutes():分钟

Dt.getSeconds():秒

Dt.getDay():星期，从0开始

Dt.toDateString():英文日期

Dt.toLocalDateString():日期

Dt.toTimeString():英文时间

Dt.toLocalTimeString():时间

Dt.valueOf():毫秒

String

.length 字符串的长度

.charAt(索引) 返回值是指定索引位置的字符串，超出索引，结果是空字符串

.concat(字符串A,字符串B,....) 返回的是拼接之后新的字符串

.indexOf(想要找的字符串,从某个位置开始的索引) 返回的是这个字符串的索引值，没有找到则返回-1

.lastIndexOf(想要找的字符串) 从后面向前找，但是索引仍然是从左向右的方式，找不到则返回-1

.replace(原来的字符转,新的字符串) 用来替换字符串

.slice(开始的索引,结束的索引) 从索引的位置开始提取，到结束索引的前一个结束，并返回提取后的字符串

.split(删除的字符串,留下的个数) 切割字符串,返回一个数组

.substr(开始索引的位置,截取个数) 返回的是截取后的新字符串

.substring(开始的索引,结束的索引) 返回截取后的字符串,不包含结束索引的字符串

.toLocaleLowerCase() 转小写

.toLowerCase() 转小写

.toLocaleUpperCase() 转大写

.toUpperCase() 转大写

.trim() 干掉字符串两端的空格

array