# javascript 基础篇

# JavaScript运算符介绍

JavaScript运算符可以分为五大类：1 算术运算符，2.赋值运算符，3关系运算符，4逻辑运算符，5其他运算符

### 赋值运算符

Javascript 赋值运算就是一等号（=）;计算右边的赋值表达式，把该值复制存储在左边变量的存储单位中;如 var x = x+1

### 算术运算符

Javasciprt 算术运算符操作一个或两个数值（文字量或者变量），返回一个数值类型的结果



前自增：运算符放在操作数的左边，先对操作数进行自增，再进行其他运算，如：x=5;y=3+(++x),结果 x=6,y = 9

后自增：运算符放在操作数的右边，先让自增数参与其他运算，再对自增数据进行自增，如： x=5;y=3+(x++);结果 x=6；y=8

### 关系运算符（比较运算符）

关系运算符有等于(=)，不等于(!=)，大于(>)，小于(<)，大于等于(>=)，小于等于(<=)

示例：



注意：1. NaN==NaN =>false

1. null == undefined => true
2. null ===undefined =>false

### 逻辑运算符

Javascript 的逻辑运算符只有三个

(1)逻辑与(&&)：当两个或多个操作数都为true,逻辑结果才为true

(2)逻辑或(||): 当两个或多个操作数其中一个为true,逻辑结果为true

(3)逻辑非:是一个一元运算符，当操作数结果为true,逻辑结果为false,否则反之

### 位运算符

Javascript 位运算符按位（指二进制单位）计算操作数得到结果



### 其他运算符

1. 三元运算符（x？a：b）;如x 为true,表达式结果为a,否则为b
2. typeof a: 返回 a 的数据类型(基本类型或者引用类型)

### 运算符的优先级





# JavaScript数据类型转换

Javascript 是一种弱类型语言，不需要很关心变量的存储类型

### 基本数据类型转换

1. String():将其他数据类型转为字符串，如：String(1)=>”1”

String({a:1})=>”"[object Object]"”,String([1,2])=>”1,2”

1. Number():将其他类型转为数字类型，只能转字符型的数字，否侧转换结果为NaN,如Number(“a”)=NaN,Number(“1”) = 1
2. Boolean(): 除了“”（空字符）,NaN,null,undefind,0,其他值都为true

### 从一个值中提取另一个值的类型得到类型转换

1. parseInt(String,numberBase?):将字符串转为整数，转换过程：从字符串第一个字符开始向右截取到非数字字符时停止，然后将结果转为整型，否则返回NaN,如：parseInt(“12aa”)=>12;parseInt(“aa12”)=NaN
2. parseFloat():用法和parseInt大致相同，不同的是转换的时候可以保留小数
3. evel(): 可以执行语句和表达式的方法(不推荐用，安全性低)

eval( "(9+5)\*2" ); //28

eval("var abc");//返回undefined

eval("abc");//报错

# JavaScript变量及变量作用域

变量用于存储数据

### 数据类型：基本类型和引用类型

1. 基本类型：null，undefined,number,string,boolean,sybool
2. 引用类型：就是object类型，比如数组，对象，函数等

### 变量的声明：let ,var,const

1. let:声明变量的值可以修改，相同作用域内只能声明一次，具有块级作用域，不存在变量提升
2. var:声明变量的值可以修改，可以重复声明，不初始化值为undefined,作用域再起执行的函数内，存在变量提升
3. const: 常量的声明，一般变量名称全为大写，声明的变量值不能修改，必须初始化，一般在顶层作用域声明

### 作用域

作用域分为：全局作用域，局部作用域(函数作用域)，块级作用域(es6)

1. 全局作用域：可以在任何地方都可以使用
2. 局部作用域：只能在当前执行函数环境中使用
3. 块级作用域：使用let 声明，在{} 大括号声明相同变量中外层和内层互不影响，在块级声明函数，必须使用表达式声明，如果使用函数式声明(具有函数提升)，会出现报错