**1数据类型**

分为原始类型和引用类型

**原始类型**——数值型、字符串、布尔型、未定义型、空

数值型——Number

分为整型和浮点型

// 整数

var n1 = 4;

// 8进制-0开头

var n2 = 010;

//十六进制=0x开头

var n3 = 0xf;

//浮点小数

var n4 = 3.1415e2;

var n5 = 3141.5e-1;

//定点小数

var n6 = 314.15;

检测数据类型——typeof 数据

字符串型——string引号不分单双

//查看任意一个字符的编码Unicode

'a'.charCodeAt()

布尔型——true false 真假

常用于记录当前状态，例如是否登录，是否注册，是否会员。。。

一些运算的结果也是布尔型的值 例如 5<3

未定义——undefined

没有保存数据

空——null——object

值为null 类型是object，常用于和对象(引用类型)数据一起使用

**2数据类型转换**

隐式转换

运算过程中自动发生的转换，不需要人为干预

数值+字符串 数字转为字符串 1+'2' //'12'

数值+布尔 布尔转数值 true—1 false—0

布尔+字符串 布尔转字符串 拼接

查看以下程序的运行结果

var a=5,b=true,c='haha'

console.log(a+b+c); //6haha

console.log(b+c+a); //truehaha5

console.log(c+a+b); //haha5true

|  |
| --- |
| 加号(+)的作用  执行数值间的加法运算  执行字符串间的拼接，会将数据转为字符串 |

- \* / 全转数值

var q1 = '5'-'3a' //NaN——Not a Number 不是一个数字

var q3 = 1 + underfined NaN

var q2= 3+unll //3————3+0

**隐式转换的特点（加减乘除）**

如果加号两端的数据有一个为字符串，则另- -个也会转为字符串，最后执行拼接。

其他情况会将非数字隐式转换为数值型，在转换过程中自动调用函数Number来完成的

强制转换为数值型

Number().

|  |
| --- |
| Number('5') //5.  Number('3a') //NaN.  Number(true) //1.  Number(false) //0  Number(null) //O  Number(undefined) //NaN. |

NaN : Not a Number,不是一个数字，没有成功转为数值返回这个结果，NaN和任何数字执行加减乘除运算结果还是NaN|

强制转为整型

通常用于将字符串和小数转为整型

parselnt( ).

|  |
| --- |
| parselnt(3.14) //3.  parselnt('6.18a') //6.  parseInt(a6.18') //NaN. |

转为整型的还是,会从数据的开头寻找整数，如果找不到返回NaN.

强制转为浮点型

常用于将字符串转为浮点型

parseFloat()

|  |
| --- |
| parseFloat('3.14") //3.14  parseFloat('6.18a') //6.18.  parseFloat('a6.18') //NaN  parseFloat('5a") //5 |

将数值和布尔型转字符串

.toString();

|  |
| --- |
| var num = 2;  var str = num.toString(); |

**3运算符**

表达式：由数据或者由运算符连接的数据所组成的形式

算数运算符、比较运算符、逻辑运算符、位运算符、赋值运算符、三目运算符

**算数运算符**

+ - / \* %取余 ++自增 --自减

|  |
| --- |
| var a=1;  // var b=a++; //先把 a的值赋给b,然后a再执行自增  // var c=++a; //先让a执行自增，然后把自增后的结果赋给C |

**比较运算符**

> < >= >= ==等于 !=不等于 ===全等于 !==不全等于

== 只是比较值是否相同，可能发生数据类型转换

=== 不仅比较值，还会比较类型

|  |
| --- |
| 在比较相等或不等推荐用=== 和!== |

3>'10' //false

字符串隐式转换为值

'3'>'10' //true

字符串比较,比较的是首个字符的Unicode码

3-> 51 1->49.

console.log(3>'10a'); //false

console.log(3<'10a'); //false

console.log(3=='10a'); //false

console.log(NaN==NaN); //false

NaN和任何值比较（> < == >= <= ===）都是false

包括NaN

**逻辑运算符**

&&并且 ||或着 !非

&& 关联的两个条件都是true,结果是true,否则是false

|| 关联的两个条件有一个是 true,结果是true,否则false

! 取反

**短路逻辑**

&& 当第一个条件为false，就不再执行第二个条件

|| 当第一个条件为true，就不再执行第二个条件

查看以下程序是否会报错

var n = 2;

n>3 && console.log(a);

n<1 || console.log(a);