布尔类型

> 只有两个值：true / false

如何把其它数据类型转换为布尔类型?

- Boolean
- !
- !!

```
1. Boolean(1) =>true
2.
3. !'珠峰培训' =>先把其它数据类型转换为布尔类型，然后取反
4.
5. !!null =>去两次反，等价于没取反，也就剩下转换为布尔类型了
```

规律：`在JS中只有“0/NaN/空字符串/null/undefined”这五个值转换为布尔类型的false，其余都转换为true`

---

null && undefined

> 都代表空或者没有
>
> - null：空对象指针
> - undefined：未定义

null一般都是意料之中的没有（通俗理解：一般都是人为手动的先赋值为null，后面的程序中我们会再次给他赋值）

```
var num = null; //=>null是手动赋值，预示着后面我会把num变量的值进行修改
...
num = 12;
```

undefined代表的没有一般都不是人为手动控制的，大部分都是浏览器自主为空（后面可以赋值也可以不赋值）

```
var num; //=>此时变量的值浏览器给分配的就是undefined
...
后面可以赋值也可以不赋值
```

刘天瑞（BOY）的女朋友是null，他的男朋友是undefined

---

object对象数据类型

> 普通对象
>
> - 由大括号包裹起来的
> - 由零到多组属性名和属性值（键值对）组成

属性是用来描述当前对象特征的，属性名是当前具备这个特征，属性值是对这个特征的描述（专业语法，属性名称为键[key]，属性值称为值[value]，一组属性名和属性值称为一组键值对）

```
var obj = {
    name:'珠峰培训',
    age:9
};
//=>对象的操作：对键值对的增删改查
语法：对象.属性 / 对象[属性]

[获取]
obj.name
obj['name']   一般来说，对象的属性名都是字符串格式的（属性值不固定，任何格式都可以）

[增/改]
JS对象中属性名是不允许重复的，是唯一的
obj.name='周啸天'; //=>原有对象中存在NAME属性，此处属于修改属性值
```

```
obj.sex='男'; //=>原有对象中不存在SEX，此处相当于给当前对象新增加一个属性SEX
obj['age']=28;

[删]
彻底删除：对象中不存在这个属性了
delete obj['age'];

假删除：并没有移除这个属性，只是让当前属性的值为空
obj.sex=null;

----
在获取属性值的时候，如果当前对象有这个属性名，则可以正常获取到值（哪怕是null），但是如果没有这个属性名，则获取的结果是undefined
obj['friends'] =>undefined
```

思考题：

```
var obj = {
    name:'珠峰培训',
    age:9
};
var name = 'zhufeng';

obj.name  =>'珠峰培训'   获取的是NAME属性
```

```
    的值
8.  obj['name'] ='珠峰培训' 获取的是NAME属
    性的值
9.  obj[name] =>此处的NAME是一个变量,我们要获
    取的属性名不叫做NAME，是NAME存储的值'zhuf
    eng' =>obj['zhufeng'] =>没有这个属性,属
    性值是undefined
10.
11. ----
12. 'name' 和 name 的区别?
13.   => 'name'是一个字符串值，它代表的是本身
14.   => name是一个变量，它代表的是本身存储的
    这个值
```

一个对象中的属性名不仅仅是字符串格式的，还有可能是数字格式的

```
1.  var obj = {
2.      name:'珠峰培训',
3.      0:100
4.  };
5.  obj[0] =>100
6.  obj['0'] =>100
7.  obj.0 =>Uncaught SyntaxError: Unexpec
    ted number
8.
9.  ----
10. 当我们存储的属性名不是字符串也不是数字的时
```

```
    候，浏览器会把这个值转换为字符串（toStrin
    g），然后再进行存储
11.
12. obj[{}]=300;  =>先把({}).toString()后
    的结果作为对象的属性名存储进来 obj['[objec
    t Object]']=300
13.
14. obj[{}] =>获取的时候也是先把对象转换为字符
    串'[object Object]',然后获取之前存储的30
    0
15.
16. ----
17. 数组对象（对象由键值对组成的）
18. var oo = {
19.     a:12
20. };
21. var ary = [12,23]; //=>12和23都是属性
    值，属性名呢？
22.
23. 通过观察结果，我们发现数组对象的属性名是数
    字（我们把数字属性名称为当前对象的索引）
24. ary[0]
25. ary['0']
26. ary.0  =>报错
```

JS中的判断操作语句

1、if / else if / else

```
var num = -6;
if(num>10){
    num++; //=>num=num+1 num+=1 在自身的基础上累加1
}else if(num>=0 && num<=10){
    num--;
}else{
    num+=2;
}
console.log(num);
```

只要有一个条件成立，后面不管是否还有成立的条件，都不在判断执行了

```
var num = 10;
if(num>5){
    num+=2;
}else if(num>8){
    num+=3;
}else{
    num+=4;
}
console.log(num); //=>12
```

关于条件可以怎么写？

```
// >= / <= / == 常规比较
if(0){
    //=>不管你在条件判断中写什么，最后总要把其计算出TRUE/FALSE来判断条件是否成立（把其它类型的值转换为布尔类型，只有 0/NaN/''/null/undefined 是false，其余都是true）
}

if('3px'+3){
    //=>在JS中，+ - * / % 都是数学运算，除 + 以外，其余运算符在运算的时候，如果遇到了非数字类型的值，首先会转换为数字类型（Number），然后再进行运算

    //=>+ 在JS中除了数学相加，还有字符串拼接的作用（如果运算中遇到了字符串，则为字符串拼接，而不是数学相加）

    '3px'+3 =>'3px3'
}
if('3px'-3){
    '3px'-3 =>NaN
}
```

BAT面试题：

```
var num = parseInt('width:35.5px');
if(num==35.5){
    alert(0);
}else if(num==35){
    alert(1);
}else if(num==NaN){
    alert(2);
}else if(typeof num=='number'){
    //=>先算typeof num
    //=>在做比较
    alert(3);//=>alert输出的都是字符串格式的 '3'
}else{
    alert(4);
}
```

typeof

> 在JS中用来检测数据类型的方式之一，除了它以外，还有：
>
> - instanceof
> - constructor
> - Object.prototype.toString.call()

```
语法：typeof [value] =>检测value的数据类
```

```
型

返回值：使用typeof检测出来的结果是一个字符串，字符串中包含着对应的数据类型，例如："number"/"string"/"boolen"/"undefined"/"object"/"function"

typeof null =>"object" 因为null代表空对象指针（没有指向任何的内存空间）

typeof检测数组/正则/对象，最后返回的都是"object"，也就是基于这种方式无法细分对象

面试题：
console.log(typeof []);
//=>"object"

console.log(typeof typeof []);
//=>typeof "object"
//=>"string"
```

2、三元运算符

> 语法：条件?成立做的事情:不成立做的事情; <=>相当于简单的if/else判断

```
var num=12;
```

```
if(num>10){
    num++;
}else{
    num--;
}
//=>改写成三元运算符
num>10?num++:num--;
```

特殊情况

```
//=>如果三元运算符中的某一部分不需要做任何的处理，我们用 null/undeifned/void 0...占位即可
var num = 12;
num>10?num++:null;

//=>如果需要执行多项操作，我们把其用小括号包裹起来，每条操作语句用逗号分隔
num=10;
num>=10?(num++,num*=10):null;
```

思考题

```
var num = 12;
if(num>0){
    if(num<10){
        num++;
```

```
    }else{
        num--;
    }
}else{
    if(num==0){
        num++;
        num=num/10;
    }
}
改写成三元运算符！
```

3、switch case

> JS中的一种判断方式

```
var num = 12;
if(num==10){
    num++;
}else if(num==5){
    num--;
}else{
    num=0;
}

//=>改成switch case
switch(num){
    case 10:
```

```
            num++;
            break;
        case 5:
            num--;
            break;
        default:
            num=0;
}

//=>switch case 应用于变量（或者表达式等）在不同值情况下的不同操作，每一种case结束后都要加break（结束整个判断）
```

switch case中每一种case情况的比较都是基于”===”绝对相等来完成的

```
'10'==10
=>true 相等比较,如果等号左右两边的类型不一样，首先会转换为一样的数据类型，然后再进行比较
=>当前案例中，就是把字符串'10'转换为数字了，然后再比较的

'10'===10 绝对比较，如果两边的数据类型不一样，则直接不相等，它要求类型和值都完全一样才会相等（真实项目中为了保证代码的严谨性，我们应该更多使用绝对比较）
```