

# JavaScript数据类型及类型转换

## 运算符



- 能够说出JavaScript的数据类型有哪些?
- 掌握数据类型检测的方法
- 掌握类型转换的方法
- 熟练掌握各种运算符的用法

- JavaScript 数据类型
- JavaScript 数据类型检测
- JavaScript 数据类型的转换
- 运算符

赋值运算符、算术运算符、+运算符、关系运算符、条件运算符、逻辑运算符、in运算符、delete运算符

## ◆ 数据类型

**可分为原始类型和对象类型，也可以分为可变类型和不可变类型。**

可变类型的值是可修改的。对象和数组属于可变类型；数字、布尔值、NULL和undefined属于不可变类型。字符串可以看成由字符组成的数组，但是是不可变的。

## ◆ 数据类型

### • 五种原始数据类型

**Number** 值是数字

**String** 值是字符串

**Boolean** 值为布尔值 只有true或false---真或假

**Undefined** 值未定义

**Null** 值为空 (空对象)

### • 一种复合数据类型 **Object**

对象object、数组Array、函数Function等都属于object类型

## ◆ 数据类型检测

数据类型检测: **typeof**(x) 或 **typeof** x

```
var a = 100;  
var b = "小明";  
console.log(typeof a);    // 输出number  
console.log(typeof b);    // 输出string
```



## ◆ String类型

**字符串是存储字符的变量。**

字符串的字面量，必须用双引号、单引号包裹起来。  
字符串被限定在同种引号之间，即必须是成对单引号或成对双引号。

```
var str1 = "今天天气很好" ;
```

必须是同种引号，下面的写法是错误的：

```
var str2 = '哈哈' ;
```

一个数字用引号包裹起来，就是字符串类型：

```
var str3 = "3" ;
```

正常情况下，双引号里面只能用单引号或者单引号里只能用双引号：

```
var str4 = "老师说你像'考拉'一样漂亮" ;
```

```
var str5 = '老师说你像"考拉"一样漂亮' ;
```

## ◆ String类型

**字符串是存储字符的变量。**

双引号内部如果要使用相同的引号，可以使用 \ 反斜杠进行转义：

```
var str6= "老师说你像\"考拉\"一样漂亮";
```

反斜杠自己也用反斜杠来转义：

```
var str7="c:\\a\\b.jpg");
```

常用的转义字符：

\n 回车换行

```
var str8="你好\n啊\n我是谁\n啊";
```



## ◆ Boolean类型

- 布尔（逻辑）只能有两个值：true 或 false。布尔值常用在条件测试中。

```
var a=true;
```

```
var b=false;
```

Boolean类型中，字符串、非 0 数字、对象，将返回true；空字符串、数字0、undefined、null、NaN，将返回 false。

## ◆ Boolean类型转换

### 把其它类型转化为Boolean的方法:

- **Boolean()** 属于强制类型转换方法

练一练:

```
var b1=Boolean( "str");
```

```
var b2=Boolean(1);
```

```
var b3=Boolean({"age":"10"});
```

```
var b4=Boolean("");
```

```
var b5=Boolean(0);
```

```
var b6=Boolean(undefined);
```

```
var b7=Boolean(null);
```

```
var b8=Boolean(NaN);
```

## ◆ Undefined类型

- 使用var声明变量但未对其赋值，这个变量就是undefined。

```
var a;
```

```
console.log(a); //undefined
```

- 声明变量以后输出和未声明变量输出区别：

```
var a;
```

```
console.log(a);           //undefined
```

```
console.log(b);           //没有声明b，报错
```

## ◆ Null类型

- 从逻辑角度看，null值表示一个空对象指针，所以typeof操作符检测null时会返回“object”。

```
var a = null;  
console.log(typeof a);    //object
```

- Undefined值派生自null，所以ECMA-262规定对它们的相等性测试要返回true。

```
console.log(null == undefined);    //true  
console.log(null === undefined);   //false
```

## ◆ Null和Undefined类型的用法（在此只做了解）

null和undefined基本是同义的，只有一些细微的差别，典型用法是：

- null表示“没有对象”，即该处不应该有值。
  - (1) 作为函数的参数，表示该函数的参数不是对象。
  - (2) 作为对象原型链的终点。
- undefined表示“缺少值”，就是此处应该有一个值，但是还没有定义。
  - (1) 变量被声明了，但没有赋值时，就等于undefined。
  - (2) 调用函数时，应该提供的参数没有提供，该参数等于undefined。
  - (3) 对象没有赋值的属性，该属性的值为undefined。
  - (4) 函数没有返回值时，默认返回undefined。

## ◆ Number类型

JavaScript 唯一的数字类型，不再细分为整型int、浮点型float等。

数字字面量：十进制、八进制和十六进制

- **十进制数**：最基本的数字

```
var a=34;  
var b=100;
```

- **八进制数**：

八进制字面值第一位必须是零，后面的数字必须是0~7之间的数，超出范围，前面的零被忽略，数值当作十进制解析。

```
var num1=070;    //八进制的56  
var num2=079;    //无效的八进制，解析为79  
var num3=08;      //无效的八进制，解析为8
```



## ◆ Number类型

### • 十六进制数:

十六进制字面值前两位必须是0x，后跟任何十六进制数字（0~9和A~F）。A~F不区分大小写。

```
var num1 = 0xA;    //十六进制的10  
var num2 = 0x1f;    //十六进制的31
```

提示：尽管所有整数都可以表示为八进制或十六进制的字面量，但所有数学运算返回的都是十进制结果。

## ◆ Number类型转换

把其它类型转化为number的方法:

- **Number ()**                      属于强制类型转换方法

把变量的值转换为数字，转换的是整个值，而不是部分值。

如果变量的值无法转换为数字，那么 Number() 函数返回 NaN。

```
//强制把其它类型转换成number类型(Number方法)
var str1="123";
var str2="abc";
var str3="123abc";
console.log(Number(str1));      //123
console.log(Number(str2));      //NaN
console.log(Number(str3));      //NaN
console.log(typeof str1);       //string
console.log(typeof Number(str1)); //number
```

## ◆ Number类型转换

### 把其它类型转化为number的方法:

- **Number ()** 属于强制类型转换方法

练一练:

Number(false)

Number(true)

Number(undefined)

Number(null)

Number( "1.2" )

Number( "12" )

Number( "1.2.3" )

Number(new Object())

Number(50)

## ◆ Number类型转换

### 把其它类型转化为number的方法:

- **parseInt** (string,radix) 可解析一个字符串，并返回一个整数。

string 必填项，表示要解析的字符串；

radix 可选项，表示要解析的数字的基数（进制数）。介于2-36之间，如果该参数小于2或者大于36，则parseInt()将返回NaN。

### 注意:

只有字符串中的第一个数字会被返回。  
开头和结尾的空格是允许的。

如果字符串的第一个字符不能被转换为数字，那么 parseInt() 会返回 NaN

```
//强制把其它类型转换成number类型(parseInt方法)
console.log(parseInt("2017 年都会顺利")); //2017
console.log(parseInt("2017 年 3 月")); //2017
console.log(parseInt("123px")); //123
console.log(parseInt("123.6")); //123
```

## ◆ Number类型转换

### 把其它类型转化为number的方法:

- **parseInt** (string,radix)

练一练:

```
parseInt(true);
```

```
parseInt( "19 ",10);
```

```
parseInt(" 11",2);
```

```
parseInt("017",8);
```

```
parseInt( "1f",16);
```

```
parseInt("010");
```

```
parseInt("He was 40");
```

```
parseInt( "40 is a number" );
```

```
parseInt( "40.4.4" );
```

只有字符串中的第一个数字会被返回;  
开头和结尾的空格是允许的;  
如果字符串的第一个字符不能被转换为数字, 那么  
parseInt() 会返回 NaN。



## ◆ Number类型转换

### 把其它类型转化为number的方法:

- **parseFloat** (string) 可解析一个字符串，并返回一个浮点数。

### 注意:

开头和结尾的空格是允许的。

如果字符串的第一个字符不能被转换为数字，那么 `parseFloat()` 会返回 NaN。

如果只想解析数字的整数部分，请使用 `parseInt()` 方法。

```
//强制把其它类型转换成number类型(parseFloat方法)
console.log(parseFloat("123.00")); //123
console.log(parseFloat("123.67.88")); //123.67
console.log(parseFloat("123")); //123
```



## ◆ Number类型转换

### 把其它类型转化为number的方法:

- **parseFloat** (string)

练一练:

```
parseFloat("10");  
parseFloat( "10.00");  
parseFloat( "10.33" );  
parseFloat("34 45 66" );  
parseFloat(" 60  ");  
parseFloat("40 years" );  
parseFloat("He was 40" );  
parseFloat( "12345red");  
parseFloat( "11.22.33");  
parseFloat( "0102");  
parseFloat( "0xA");  
parseFloat( "red");
```

开头和结尾的空格是允许的。

如果字符串的第一个字符不能被转换为数字，  
那么 parseFloat() 会返回 NaN。

如果只想解析数字的整数部分，请使用 parseInt() 方法。

## ◆ Number类型转换

- 什么情况下会产生NaN?

NaN(Not a Number的缩写) 是number类型里的特殊值, 当做数学运算失败的时候, 或当其它数据类型转化为number类型失败的时候, 会得到NaN的结果。

- `isNaN()` 用于检查变量是否是非数字值。

如果把NaN 与任何值 (包括其自身) 相比得到的结果均是 `false`, 所以要判断某个值是否是 NaN, 不能使用 `==` 或 `===` 运算符。正因为如此, `isNaN()` 函数是必需的。

```
// isNaN返回的是true和false
// 如果不是一个数字, 返回true, 是一个数字返回false
// NaN不能与任何值进行比较, 包括它本身
console.log(isNaN(123)); //false
console.log(isNaN(Number("123hello"))); //true
console.log(isNaN(5-2)); //false
console.log(NaN===NaN); //false
```

## ◆ Number类型转换

- isNaN()

练一练:

```
isNaN(123.12);
```

```
isNaN(-1.23);
```

```
isNaN(5-7);
```

```
isNaN(0);
```

```
isNaN("Hello");
```

```
isNaN( "2005/12" );
```

```
isNaN(parseInt( "2005/12" ));
```

## ◆ Object类型

### • JavaScript对象

对象由花括号分隔。在括号内部，对象的属性以名称和值对的形式 (name : value) 来定义。属性由逗号分隔：

```
var obj={name: "小明" , age: "20" };
```

空格和折行无关紧要。声明可横跨多行：

```
var obj={  
  name: "zeng ",  
  age: "ni "  
};
```

对象属性调用：**obj.属性名称**

```
console.log(obj.name);
```

## ◆ Object类型

### • JavaScript数组

数组由中括号分隔。在中括号内部，数组的元素由逗号分隔：

```
var arr =[" 30"," 20", "10"];
```

数组下标是基于零的，所以第一个的下标是 [0]，第二个是 [1]，以此类推。

数组元素的调用：**arr[数组下标]**

获得数组里的第一个元素：

```
console.log(arr[0])
```



## ◆ Object类型

### • 原始类型与Object类型有着根本区别

1. 原始值是不可更改的。原始值的比较是值的比较，只有它们的值相等时才相等

2. 对象是可变的，值是可修改的：

```
var obj={x:1};//定义一个对象
```

```
obj.x=2;//对象中属性x的值更改为2
```

```
obj.y=3;//增加新属性y
```

3. 两个对象的比较并非值的比较，即使同样的属性和值，也是不相等。索引完全相等的两个数组也不等：

```
var obj={x:1},obj1={x:1}; //具有相同属性的两个对象
```

```
console.log(obj===obj1); //false,两个单独的对象永不相等
```

```
var a=[],b=[]; //两个单独的空数组
```

```
console.log(a===b); //false,两个单独的数组永不相等
```



## ◆ Object类型

### • 对象的比较

通常将对象称为引用类型，对象的比较均是引用的比较，当且仅当它们引用同一个对象时，才相等。

```
var arr=[];           //定义一个引用空数组的变量arr
var arr1=arr;         //变量arr2引用同一个数组
arr1[0]=1;            //通过变量b来修改引用的数组
console.log(arr1[0]);  //1
console.log(arr[0]);   //1，变量arr也会修改
console.log(arr===arr1); //true，arr和arr1引用同一个数组，所以相等
```

## ◆ 运算符

运算符也可以叫做操作符。

主要学习以下运算符：

- 赋值运算符
- 算术运算符
- +运算符
- 关系运算符
- 条件运算符
- 逻辑运算符
- in运算符
- delete运算符

## ◆ 赋值运算符

=

赋值运算符并不是等于

如果我想把5这个值赋值给变量a

则：var a=5;

## ◆ 赋值运算符

给定  $x=10$  和  $y=5$ , 下面的表格解释了赋值运算符:

运算符	例子	等价于	结果
=	$x=y$		$x=5$
+=	$x+=y$	$x=x+y$	$x=15$
-=	$x-=y$	$x=x-y$	$x=5$
*=	$x*=y$	$x=x*y$	$x=50$
/=	$x/=y$	$x=x/y$	$x=2$
%=	$x\%=y$	$x=x\%y$	$x=0$

## ◆ 算术运算符

算术运算符用于执行变量与/或值之间的算术运算。

加

+

减

-

乘

\*

除

/

求模/取余

%

## ◆ 算术运算符

给定 $y=5$ ，下面的表格解释了算术运算符：

运算符	描述	例子	结果
+	加	$x=y+2$	$x=7$
-	减	$x=y-2$	$x=3$
*	乘	$x=y*2$	$x=10$
/	除	$x=y/2$	$x=2.5$
%	求余数	$x=y\%2$	$x=1$
++	累加	$x=++y$	$x=6$
--	递减	$x=- -y$	$x=4$



## ◆ 算术运算符

求模/取余      %

例：  $5\%3$

语言描述：5对3取余

计算方法：5除以3所得到的余数

$5\%3=2$

$$\begin{array}{r} 1 \\ 3 \overline{) 5} \\ \underline{3} \\ 2 \end{array}$$

## ◆ 算术运算符

求模/取余      %

例： 135%9

语言描述：135对9取余

计算方法：135除以9所得到的余数

$135\%9=0$

$$\begin{array}{r} 15 \\ 9 \overline{) 135} \\ \underline{9} \phantom{0} \\ 45 \\ \underline{45} \\ 0 \end{array}$$

## ◆ 算术运算符取余思考

$$2\%7 \quad 2$$

$$(-2) \% 7 \quad -2$$

$$2\% (-7) \quad 2$$

$$(-2) \% (-7) \quad -2$$

$$7\%4 \quad 3$$

$$(-7) \% 4 \quad -3$$

$$7\% (-4) \quad 3$$

$$(-7) \% (-4) \quad -3$$

余数符号与被除数符号一致（跟分子走）

## ◆ 递增和递减

++  
--

i++  
i--

++i  
--i

++表示的是变量的值加1

++i表示的是，先加1再执行

而i++表示的是先执行再加1

```
var i=4;
```

```
var k;
```

```
k=i++;
```

k= ?    i= ?

```
var i=4;
```

```
var k;
```

```
k=++i;
```

k= ?    i= ?

## ◆ +运算符

用于把文本值或字符串变量连接起来。

如需把两个或多个字符串变量连接起来，请使用 + 运算符。

```
var str1 = "今天是个";
```

```
var str2 = "好天气!";
```

```
var str3 = str1 + str2;
```

```
console.log(str3);
```

**str3的结果是：**今天是个好天气!

## ◆ +运算符

对字符串和数字进行拼接

```
var x=5+5;  
console.log(x);    //10  
var x="5"+"5";  
console.log(x);    //55  
var x=5+"5";  
console.log(x);    //55  
var x="5"+5;  
console.log(x);    //55
```

**如果把数字与字符串相加，  
结果将成为字符串。**



## ◆ +运算符

+运算符的隐式转换规则:

- 字符串与数字进行+拼接时, 数字会隐式的转换成字符串
- 字符串与对象进行+拼接时, 对象会隐式的转换成字符串
- 数字与null进行+拼接时, null会转换成数字类型
- 布尔值之间进行+拼接时, 布尔值会转换成数字类型
- null和undefined和数字进行拼接时, 会转换成数字类型, null为0, undefined为NaN
- 字符串与数字进行-号运算时, 字符串会隐式的转换成数字类型

## ◆ +运算符

### 练一练

10 + 10

"10" + 10

"10" - 10

1+{ }

true+true

2+null

2+undefined

## ◆ 面试题思考

```
var str= "123abc" ;  
console.log(typeof str);  
console.log(str++);  
console.log(typeof str++);
```

```
var str= "123abc" ;  
console.log(typeof str);  
console.log(str+1);  
console.log(typeof (str+1));
```

## ◆ 关系运算符（比较运算符）

关系操作符在逻辑语句中使用，以测定变量或值是否相等。

大于	小于	等于	小于等于	大于等于	不等于
>	<	<div><div>==</div><div>===</div></div>	<=	>=	!=

等于（==）的情况下 只要值相同就返回True

全等（===）的时候需要值和类型都要匹配才能返回True

关系操作符返回的是布尔值 true 或 false

## ◆ 关系运算符（比较运算符）

给定 $x=5$ ，下面的表格解释了关系运算符：

运算符	描述	例子	结果
<code>==</code>	等于	<code>x==8</code>	false
<code>===</code>	全等(值和类型)	<code>x===5</code> <code>x=== "5"</code>	true false
<code>!=</code>	不等于	<code>x!=8</code>	true
<code>&gt;</code>	大于	<code>x&gt;8</code>	false
<code>&lt;</code>	小于	<code>x&lt;8</code>	true
<code>&gt;=</code>	大于或等于	<code>x&gt;=8</code>	false
<code>&lt;=</code>	小于或等于	<code>x&lt;=8</code>	false

## ◆ 关系运算符（比较运算符）

- 纯数字之间比较

```
console.log(1 < 3); // true
```

- 数字字符串比较，转换成ASCII码比较

```
console.log("1" < "3"); // true
```

```
console.log("123" < "123"); // false
```

```
console.log("123" < "1234"); // true
```

- 纯字符串比较,先转成ASCII码

```
console.log("j" < "k"); // true
```

```
console.log("abc" < "aad"); // false, 多纯字母比较, 会依次比较ASCII码
```



## ◆ 关系运算符（比较运算符）

- 汉字比较，转成ASCII码

```
console.log("我".charCodeAt());//25105
```

```
console.log( "的" .charCodeAt());//30340
```

```
console.log( "我" < "的");//true
```

- 当数字和字符串比较，且字符串为数字。则将数字字符串转为数字

```
console.log(123 < "124");//true
```

- 当数字和字符串比较,且字符串为非纯数字时,则将非数字字符串转成数字的时候会转换为NaN,当NaN和数字比较时不论大小都返回false.

```
console.log(13 > "abc");//false
```

## ◆ 关系运算符（比较运算符）

### 练一练

var a = "5"	a >= b	true
var b = 5	a == b	true
var c = 15	a === b	false
var d = "15"	a < d	false
var e = "abc"	b < c	true
var f = "abbb"	e < f	false
	e < a	false

## ◆ 条件运算符（三元操作符）

表达式1 **?** 表达式2 **:** 表达式3

进行表达式1的判断

如果表达式1成立  
执行表达式2

如果表达式1不成立  
执行表达式3

```
var a=5,b=4;  
console.log(a>b?"ok":"no");
```

## ◆ 逻辑运算符

用于把文本值或字符串变量连接起来。

与	&&	&&前后两个均为真才可以
或		前后有一个为真就可以
非	!	! 求当前的值的非

给定  $x=6$  和  $y=3$ , 下面的表格解释了逻辑运算符:

运算符	描述	例子	结果
&&	and	$(x < 10 \&\& y > 1)$	true
	or	$(x == 5    y == 5)$	false
!	not	$!(x == y)$	true

## ◆ 逻辑运算符

练一练:

true && false  
true || false  
!true

true && true  
true || true

false && false  
false || false  
!false

## ◆ 逻辑运算符

- 逻辑与的运算规则

- 1.两边条件都为true时，结果才为true;
- 2.如果有一个为false，结果就为false;
- 3.当第一个条件为false时，就不再判断后面的条件

**注意：**当数值参与逻辑与运算时，结果为true，那么会返回的会是第二个为真的值；如果结果为false，返回的会是第一个为假的值。

```
console.log( 5 && 4 );//当结果为真时，返回第二个为真的值4  
console.log( 0 && 4 );//当结果为假时，返回第一个为假的值0
```



## ◆ 逻辑运算符

### • 逻辑或的运算规则

字符串、非 0 数字、对象，将返回true；空字符串、数字0、undefined、null、NaN，将返回 false。

- 1.只要有一个条件为true时，结果就为true；
- 2.当两个条件都为false时，结果才为false；
- 3.当一个条件为true时，后面的条件不再判断

**注意：**当数值参与逻辑或运算时，结果为true，会返回第一个为真的值；如果结果为false，会返回第二个为假的值；

```
console.log( 5 || 4 );//当结果为真时，返回第一个为真的值5
```

```
console.log( 0 || 0 );//当结果为假时，返回第二个为假的值0
```

## ◆ 逻辑运算符

- && 优先级高于 ||

```
console.log(3||2&&5||0);    //3
```

先算2&&5的值为5，然后再3||5----3，最后再3||0----3

练一练：

```
console.log( "" || null || 3 || 4);
```

```
console.log((1 && 3 || 0 )&& 4);
```

```
console.log(1 && 3 || 0 && 4);
```

```
console.log(0 && 3 || 1 && 4);
```

## ◆ 逻辑运算符

### • 逻辑非

首先会将他的操作数转化为一个布尔值，然后求反

- 如果操作数是一个对象，返回 `false`;
- 如果操作数是一个空字符串，返回 `true`;
- 如果操作数是一个非空字符串，返回 `false`;
- 如果操作数是数值 0，返回 `true`;
- 如果操作数是任意非 0 数值（包括 `Infinity`），返回 `false`;
- 如果操作数是 `null`，返回 `true`;
- 如果操作数是 `NaN`，返回 `true`;
- 如果操作数是 `undefined`，返回 `true`。

练一练：

`! "blue"`

`! 0`

`! NaN`

`! ""`

`! 123456`

## ◆ 运算符优先级

逻辑非

算术操作符

关系操作符

逻辑与 逻辑或

条件操作符

赋值操作符

## ◆ 运算符优先级

$b+e<d$      $b+(e<d)$  ?     $(b+e)<d$

算术 +  $\xrightarrow{\text{高于}}$  关系 <

$z = 78 * (96 + 3 + 45)$

圆括号可用来改变运算符优先级所决定的求值顺序。

根据运算符优先级的规则，它们将按下面的顺序求值：(), +, +, \*, =

## ◆ 运算符优先级

小括号

中括号

大括号

在数学中，我们这样书写：

$$y=(x+2)*\{[(4-x)*3-8]/4+3\}$$

在Javascript中，我们这样书写：

$$y=(x+2)*(((4-x)*3-8)/4+3)$$

小括号    提升优先级

中括号    数组

大括号    对象



## ◆ 运算符综合练习

判断变量是不是闰年

年份能够被4整除且不能被100整除, 或者能够被4\100\400整除

年份%数字==0

逻辑 && 与

逻辑 || 或

假设:

```
var year=2000;
```

写出表达式

## ◆ in运算符

如果右侧的对象拥有一个名为左操作数值的属性名，那么表达式返回true

**var obj={x:1,y:1};//定义一个对象**

console.log( "x" in obj);        //true, 对象有一个名为 "x" 的属性

console.log( "z" in obj);        //false,对象中不存在名为 "z" 的属性

**var arr=[7,8,9];//定义一个拥有三个元素的数组**

console.log(0 in arr);        //true,数组包含索引为0的元素 (arr[0])

console.log(1 in arr);        //true,数组包含索引为1的元素 (arr[1])

console.log(3 in arr);        //false,没有索引为3的元素

## ◆ delete运算符

用来删除对象属性或者数组元素。不能删除用var定义的变量

```
var obj={x:1,y:1};           //定义一个对象
delete obj.x;                 //删除一个属性
console.log("x" in obj);      //false,属性在对象中不再存在

var arr=[7,8,9];              //定义一个拥有三个元素的数组
delete arr[2];                 //删除最后一个数组元素
console.log(2 in arr);         //false,索引为2的元素不存在了

var str= "你好" ;             //用var定义一个变量
delete str;                    //删除变量str
console.log(str);              //不能删除var定义的变量
```

