1.数据类型

分为原始类型和引用类型

原始类型分为数值型、字符串型、布尔型、未定义型、空

(1)数值型

分为整型和浮点型

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

8进制

1 2 3 ... 7 10 11 12

16进制

1 2 3 .... 9 a ... f 10

8进制：以0开头的数字

16进制：以0x开头的数字，a~f代表10~16，不区分大小写

3141.5e-1

314.15

31.415e+1

3.1415e+2

浮点型分为浮点小数和定点小数

|  |
| --- |
| typeof 数据 检测数据类型 |

(2)字符串型

用引号所包含的数据，不分单双引号

查看任意一个字符的Unicode码

's'.charCodeAt()

(3)布尔型

只有两个值，分别是true和false，代表真和假

常用于记录当前的状态，例如是否登录，是否注册，是否为会员...

一些运算的结果也是布尔型的值 例如 3>5

(4)未定义型

只有一个值 undefined，声明了变量未赋值则为undefined

(5)空

只有一个值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

1)隐式转换

运算过程中自动发生的转换，不需要人为干预

(1)数值+字符串 数值转为字符串

1+'2' //'12'

(2)数值+布尔型 布尔型转为数值型 true-1 false-0

2+true //3

2+false //2

(3)布尔型+字符串 布尔型转为字符串

true+'4' //'true4'

练习：查看以下程序的运行结果

var a=5,b=true,c='tedu';

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

console.log(b+c+a); //'truetedu5'

console.log(c+a+b); //'tedu5true'

|  |
| --- |
| **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 |

**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，则另一个也会转为字符串，最后执行拼接。

其它情况会将非数字隐式转换为数值型，转换的过程中自动调用函数Number来完成的。

2)强制转换

(1)强制转换为数值型

Number()

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

(2)强制转为整型

通常用于将字符串和小数转为整型

parseInt( )

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

转为整型，会从数据的开头寻找整数，如果找不到返回NaN

(3)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 |

(4)将数值和布尔型转字符串

var num=2;

var str=num.**toString()**; // '2'

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

运算符包含有算术运算符、比较运算符、逻辑运算符、位运算符、赋值运算符、三目运算符

(1)算术运算符

+ - \* / % ++ --

% 取余

++ 自增，在原来基础之上加1

-- 自减，在原来基础之上减1

|  |
| --- |
| 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 |

练习：查看以下程序运行结果

var num=6;

var n1=num--; //num==>n1=6,num--==>num=5,n2=num-1=4.

var n2=--num; //

console.log(n1+n2); 等价于 console.log(num-- + --num)

(2)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

|  |
| --- |
| 在比较相等或者不等推荐使用 === 和 !== |

练习：

3>'10' //false

字符串隐式转换为数值

'3'>'10' //true

字符串比较，比较的是首个字符的Unicode码

3->51 1->49

3>'10a' //false

3<'10a' //false

3=='10a' //false

'10a'转为NaN，NaN和任何值比较（> < == >= <= ===）都是false

NaN==NaN //false

(3)逻辑运算符

&& (逻辑与/并且) 关联的两个条件都是true，结果是true，否则是false

|| (逻辑或/或者) 关联的两个条件有一个是true，结果是true，否则false

! (逻辑非) 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

练习：查看以下程序是否会报错

var n=2;

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

n<1 || console.log(b); //b is not defined

课后任务

(1)复习今天内容，整理思维导图

(2)练习

声明变量保存任意一个年份，如果是闰年，打印出'闰年' （使用短路逻辑）

闰年判断：能被4整除，并且不能被100整除，或者能被400整除

(3)预习第3天内容 if if-else if-else嵌套

http://www.codece.com/archives/193