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

整型

10进制

1 2 3 .... 7 8 9 10 11 ... 15 16

8进制

1 2 3 ... 7 10 11 12

16进制

1 2 3 ... 7 8 9 a ... f 10

8进制以0开头的数字，例如012

16进制以0x开头的数字，a~f代表10~15，不区分大小写

浮点型

3141.5E-1

314.15

31.415E+1

3.1415E+2

|  |
| --- |
| typeof 检测数据类型 |

(2)字符串型

被引号包含的数据，不区分单双引号

查看任意一个字符的Unicode码

'a'.charCodeAt()

(3)布尔型

只有两个值，分别是true和false，代表真和假

通常用于保存只有两个状态的数据，例如是否登录、是否在线... 一些运算符的结果也是布尔型

(4)未定义型

只有一个值是undefined，代表一个空值，例如声明了变量未赋值为undefined

(5)空

只有一个值是null，类型是object，常结合对象一起使用

2.数据类型转换

分为隐式转换和强制转换

1)隐式转换

在运算过程中自动产生的数据转换

(1)数字+字符串 数字转换为字符串

1 + '2' // '12'

(2)数字+布尔型 布尔型转换数值 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 + '5' //'true5'

|  |
| --- |
| **加号(+)的作用**  执行数字之间的加法运算  执行字符串之间的拼接 |

练习：查看以下程序的运行结果

var a = 3, b = true, c = 'tedu';

console.log(a+b+c); //'4tedu'

console.log(b+c+a);//'truetedu3'

console.log(c+a+b);//'tedu3true'

|  |
| --- |
| NaN：Not a Number(不是一个数字)  在将一个数据转换为数值型时候，没有成功得到一个数字就会返回NaN  NaN和任意数字执行运算，结果还是NaN |

所有的**隐式转换为数字**都是会自动调用函数Number完成的

2)强制转换

(1)强制转换为数值型

Number()

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

(2)强制转换为整型

parseInt()

强制将字符串和小数转为整型

|  |
| --- |
| parseInt('3.94') //3  parseInt('6.18a') //6  parseInt('a6.18') //NaN  parseInt(5.9) //5 |

undefined null true false都是返回NaN

(3)强制转换为浮点型

parseFloat()

强制将字符串转换为浮点型

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

(4)数值和布尔型强制转字符串

toString()

|  |
| --- |
| var num = 5;  num.toString() //'5' |

3.运算符

表达式：由数据本身或者由运算符连接的操作数据组成的形式称作表达式

运算符分为算术运算符、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，在原来的基础之上加1

-- 自减，在原来的基础之上减1

(2)比较运算符

> < >= <= ==(等于) !=(不等于) ===(全等于) !==(不全等于)

等于：只是比较两者的值是否相同，可能会发生隐式转换

全等于：同时比较类型和值，都相同结果才是true，否则false

不等于：比较值是否不相同

不全等于：同时比较类型和值，有一个不等为true，否则false

3>'10' 数字和字符串比较，字符串转为数值

'3'>'10' 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

|  |
| --- |
| NaN和任何值比较（> < >= <= == ===）结果都是false  NaN == NaN //false |

(3)逻辑运算符

&& || !

&& 逻辑与，关联的两个条件都为true，结果是true，否则false

|| 逻辑或，关联的两个有一个为true，结果是true，否则false

! 逻辑非，取反

**短路逻辑**

逻辑与，如果第一个条件是false，就不再执行第二个条件

逻辑或，如果第一个条件是true，就不再执行第二个条件

练习：以下程序运行是否会报错

|  |
| --- |
| var a = 3;  a > 1 && console.log(num);  a < 5 || console.log(num); |

短路逻辑关注点在于是否会执行第二个表达式。

练习：声明变量保存一个人的年龄，使用短路逻辑，如果满18岁打印‘成年人’

(4)位运算符

模拟计算机底层的运算，先把数据转为2进制，然后进行运算；当运算完以后再把结果转回成10进制。

1 2 3 4 5 6 7 8

1 10 11 100 101 110 111 1000

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，使用短路逻辑，如果是闰年打印'闰年'

能被4整除 并且不能被100整除 或者被400整除

2020 2100 2000

(3)预习js第3天 ，if if-else if-else嵌套...

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