1. 数据类型

数据分为 原始类型 和 引用类型（对象）

原始类型分为：数值型、字符串型、布尔型、未定义型（undefine）、空（null）

监测数据类型：consolo.log(typeof A)

1. 数值型

分为整型（4字节）和浮点型（8字节）

整型

JS打印的结果最终结果都是十进制

JS中以0开头的数字都为八进制（例012打印结果为10）

以0x（零 X 从10开始变成abcdef，不区大小写）开头为十六进制

浮点型

浮点型分为定点小数和指数型小数

3140=3.14e3

0.314=3.14e-1=314e-3

1. 字符串型

数据被引号所包含就是字符串型，不区分单双引号

查看某个字符的unicode码：

console.log('然'.charCodeAt()); //28982里面是.

1. 布尔型

true/false 注意小写

在程序中表示真或者假，可用于一些结果比较

1. 未定义型

声明了变量未赋值，结果 唯一值 undifined

1. 空

只有一个值null，将来会和引用类型一起使用，null默认函数类型为对象型 ，object。

1. **数据类型转换**
2. 隐式转换

在运算的过程中，自动的发生转换

1. 数字+字符串 数字 → 字符串

2+‘3’=‘23’

1. 数字+布尔型 布尔型 → 数字 true=1 false=0
2. 布尔型+字符串型 布尔型 → 字符串型

|  |
| --- |
| JS中加号（+）的作用：  执行加法运算  执行字符串间的拼接 |

例：var name='MISS shen',sex='女';

console.log('我的名字是'+name+',我的性别是'+sex);

|  |
| --- |
| 减法、乘法、除法的隐式转换（— \* /）  将运算符两端的数据转换为数值型，如果转换失败，返回NaN（Not a Number），不是一个数字，任何值与NaN执行加减乘除都会返回NaN，自动调用了Number，转换为数值型。NaN与任何数比较都为false。NaN函数类型默认为number |

特殊值：Number( 'a2' ) 输出结果为NaN，但数据类型为Number

1. 未定义型进行运算

undefined 与数值型 undefined→NaN (自调用Number)

undefined 与字符型 undefined→字符型

1. 强制转换
2. 强制转为数值型（number）

Number( );

例：Number('2a') -- > NaN

Number(undefined) -- > NaN

Number(null) -- > 0

Number('') -- > 0

任何不只是数字的字符串用number后都输出NaN.

1. 强制转换为整型（int）

parseInt( );

例：parseInt('2a'); -- >2

parseInt('a2'); -- >NaN

未定义型，空，布尔都返回NaN。如果要转换的字符串开头是非数字也返回NaN。

1. 强制转化为浮点型（基本同上）

parseFloat（ ）

1. 强制将数值型和布尔型转为字符串类型

num.toString(); (注意是。号)

1. **运算符**

表达式；由运算符连接的操作数据，所组成的形式

运算符分为算数，比较，逻辑，位，赋值，三目运算符

1. 算数运算符

+ - \* / % ++ --

%: 取余数 一般用在区分奇偶数

++ 自增 在原来基础上加1，但是要明白++是在当前操作后，内存里＋1，并不是当时就＋1

例1： var a=1; ↓这个步骤打印完后执行了自增

console.log(a,a++,a) ; = 1,1,2

console.log(a++)，先打印a的值再执行自增，

console.log(++a)，先执行自增，再打印a的值

例2： var a = 3;

console.log(a-- + --a); = 4 （3+1）

-- 自减 在原基础上减1（和上面同理）

1. 比较运算符（请注意要用==）

> < >= <= == != ===(全等于) !==(全不等于)

比较运算符返回 bool类型的值

== 和 ===的区别：

例1： console.log(7=='7'); true

console.log(7==='7'); false

==：比较两个值是否相等

===：不仅比较值，还会比较类型是否相等

如果数据类型不同，也会发生数据类型转换。

例2：console.log(3>'10'); false （隐式转换，自动用了number函数）

console.log('3'>'10'); true （字符串类比较大小是比较首字符Unicode码）

ps：查看某个字符的unicode码：

console.log('嘿'.charCodeAt());

|  |
| --- |
| NaN和任何值比较（> < == >= <= ===）,包括自身，都返回false |

只有!= 和 !== 会返回true

例3：console.log(3 != '10a'); ---> 3!= NaN true

1. 逻辑运算符**（这里的符号用在括号里面条件并存不是下面的）**

&& 并且 两个条件要同时满足，结果为true

|| 或者 两个条件满足一个，结果为true，否则false

! 取反 !true --> false 非真为假

例：console.log(2004%4==0&&2004%100==0); false

但是很有意思的是console.log( 3 && 2 ); 输出2，因为内部不会进行计算（没有==用来将其变成bool） 0在条件位上可看做false，但是并不会影响其值console.log(0&&2);输出0

**逻辑短路**：无需关注整个逻辑运算的结果，重点在是否会执行第二个条件

&& 当第一个条件为true时 执行第二个条件

|| 当第一个条件为true时就不再执行第二个条件

|  |
| --- |
| var num = 3;  num > 5 && console.log(a); 不执行第二个语句  num < 1 || console.log(a); a 执行了第二个语句 |

使用举例：

若满18岁则打印成年人

var age = 20;

agr >=18 && console.log('成年人'); //（前面不要写console写条件就行了）

例：如果为闰年则输出‘闰年’：

var a = 1996;

((a%4==0||a%400==0)&&a%100!=0) && console.log('闰年');

1. 位运算符（了解）

计算前将所有输入目标转换为二进制，执行完位运算后再转换为十进制输出。

按位与（&） 要求上下两位都为1，结果为1，否则为0

按位或（|） 上下两位包含有1，结果为1，否则为0

按位异或（^） 上下两位不同为1，相同为0

按位右移（>>）删除二进制的最后一位

按位左移（<<）在最后一位补0（会导致数字成倍增加）

按位非（！）

运算符优先级， 先执行并且，后执行或者

1. 赋值

赋值运算符：= += -= \*= /= %=

var a = 4;

a+=6; <==> a = a+6

1. 三目运算符

两个运算符连接三个数据

表达式条件? 表达式1 : 表达式2; （表达式1后不能加分号）

结果为true or false.

若条件表达式为true，则执行表达式1

若条件表达式为false，则执行表达式2

例：根据年龄判断是否成年

var age = 19;

age >= 18? console.log('成年人') : console.log('未成年人');