数值型：Number

在JS中所有的数值都是 Number 类型，包括整数和浮点数（小数）。

```
var a = 100; // 定义一个变量 a，并且赋值整数100
console.log(typeof a); // 输出变量 a 的类型

var b = 12.3; // 定义一个变量 b，并且赋值浮点数 12.3
console.log(typeof b);
```

上方代码的输出结果为：

```
number

number
```

再次补充：在 JS 中，只要是数，就是 Number 数值型的。无论整数、浮点数（即小数）、无论大小、无论正负，都是 Number 类型的。

数值范围

由于内存的限制，ECMAScript 并不能保存世界上所有的数值。

- 最大值：`Number.MAX_VALUE`，这个值为： 1.7976931348623157e+308
- 最小值：`Number.MIN_VALUE`，这个值为： 5e-324

如果使用 Number 表示的变量超过了最大值，则会返回Infinity。

- 无穷大（正无穷）：Infinity
- 无穷小（负无穷）：-Infinity

注意：`typeof Infinity`的返回结果是number。

```
var a = Infinity;
console.log(typeof a);
```

NaN

NaN：是一个特殊的数字，表示Not a Number，非数值。比如：

```
console.log("abc" / 18);  //结果是NaN

console.log("abc" * "abcd"); //按理说，字符串相乘是没有结果的，但如果你非要让JS去算，它就一定会给你一个结果。结果是NaN
```

注意：`typeof NaN`的返回结果是 number。

Undefined和任何数值计算的结果为 NaN。NaN 与任何值都不相等，包括 NaN 本身。

连字符和加号的区别

键盘上的`+`可能是连字符，也可能是数字的加号。如下：

```
console.log("吃" + "饭" + "没"); //连字符，把三个独立的汉字，连接在一起了
console.log("吃+饭+没");      //原样输出
console.log(1+2+3);        //输出6
```

输出：

```
吃饭没
吃+饭+没
6
```

总结：如果加号两边**都是** Number 类型，此时是数字相加。否则，就是连字符（用来连接字符串）。

提问："吃+饭+没"中的+是连字符还是加号？

举例1：

```
var a = "1";
var b = 2;
console.log(a + b);
```

控制台输出：

```
12
```

举例2：

```
var a = 1;
var b = 2;
console.log("a" + b); //"a"就不是变量了！所以就是"a"+2 输出a2
```

控制台输出：

```
a2
```

于是我们明白了，在变量中加入字符串进行拼接，可以被同化为字符串。【重要】

隐式转换

我们知道，`"2"+1`得到的结果其实是字符串，但是`"2"-1`得到的结果却是数值1，这是因为计算机自动帮我们进行了"**隐式转换**"。

也就是说，`-`、`*`、`/`、`%`这几个符号会自动进行隐式转换。例如：

```
var a = "4" + 3 - 6;
console.log(a);
```

输出结果：

```
37
```

虽然程序可以对`-`、`*`、`/`、`%``这几个符号自动进行"隐式转换"；但作为程序员，我们最好自己完成转换，方便程序的可读性。

浮点数的运算

运算精度问题

在JS中，整数的运算**基本**可以保证精确；但是**小数的运算，可能会得到一个不精确的结果。**所以，千万不要使用JS进行对精确度要求比较高的运算。

如下：

```
var a = 0.1 + 0.2;
console.log(a);  //打印结果：0.30000000000000004
```

上方代码中，打印结果并不是0.3，而是0.30000000000000004。

这是因为，计算机在做运算时，所有的运算都要转换成二进制去计算。然而，有些数字转换成二进制之后，无法精确表示。比如说，0.1和0.2转换成二进制之后，是无穷的，因此存在浮点数的计算不精确的问题。

处理数学运算的精度问题

如果只是一些简单的精度问题，可以使用 `toFix()` 方法进行小数的截取。

在实际开发中，关于浮点数计算的精度问题，往往比较复杂。市面上有很多针对数学运算的开源库，比如decimal.js、 Math.js。这些开源库都比较成熟，我们可以直接拿来用。

- Math.js：属于很全面的运算库，文件很大，压缩后的文件就有500kb。如果你的项目涉及到大型的复杂运算，可以使用 Math.js。
- decimal.js：属于轻量的运算库，压缩后的文件只有32kb。大多数项目的数学运算，使用 decimal.js 足够了。

在使用这几个开源库时，既可以用 cdn 的方式引入，也可以用 npm 包的方式引入。

比如说，通过 cdn 引入 decimal.js 时，可以这样用：

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>Document</title>
    </head>
    <body>
        <script
src="https://cdn.bootcdn.net/ajax/libs/decimal.js/10.2.0/decimal.min.js">
</script>
        <script>
            console.log('加法：');
            var a = 0.1;
            var b = 0.2;
            console.log(a + b);
```

```
            console.log(new Decimal(a).add(new Decimal(b)).toNumber());

            console.log('减法：');
            var a = 1.0;
            var b = 0.7;
            console.log(a - b);
            console.log(new Decimal(a).sub(new Decimal(b)).toNumber());

            console.log('乘法：');
            var a = 1.01;
            var b = 1.003;
            console.log(a * b);
            console.log(new Decimal(a).mul(new Decimal(b)).toNumber());

            console.log('除法：');
            var a = 0.029;
            var b = 10;
            console.log(a / b);
            console.log(new Decimal(a).div(new Decimal(b)).toNumber());
        </script>
    </body>
</html>
```

打印结果：

```
加法：
0.30000000000000004
0.3

减法：
0.30000000000000004
0.3

乘法：
1.0130299999999999
1.01303

除法：
0.0029000000000000002
0.0029
```

变量值的传递（赋值）

语句：

```
a = b;
```

把b的值赋给a，b不变。

来做几个题目。

举例1：

```
var a = 1;              //1
var b = 2;              //1     2
var c = 3;              //1     2      3
a = b + c;              //5     2      3
b = c - a;              //-2    3      5
c = a * b;              //-10   5     -2
console.log(a);
console.log(b);
console.log(c);
```

输出：

```
5
-2
-10
```

举例2：

```
var a = 1;
var b = 2;
var c = 3;              //1     2     3
a = a + b;              //3     1     2
b = b + a;              //5     2     3
c = c + b;              //8     3     5
console.log(a);  //3
console.log(b);  //5
console.log(c);  //8
```

输出：

```
3
5
8
```

举例3：

```
                    // a      b
var a = "1";
var b = 2;          //"1"     2
a = a + b;          //"12"    2
b = b + a;          //"12"    "212"
console.log(a);     //输出12
console.log(b);     //输出212
```

输出：

```
12
212
```

举例4：

```
                // a         b
var a = "1";
var b = 2;
a = b + a;       //"21"       2
b = b + a;       //"21"       "221"
console.log(a);  //21
console.log(b)   //221
```

效果：

```
21
221
```

举例5：（这个例子比较特殊，字符串减去数字）

```
var a = "3";
var b = 2;
console.log(a-b);
```

效果：（注意，字符串 - 数值 = 数值）

```
1
```