变量的数据类型

为什么需要数据类型

在计算机中，不同的数据所需占用的存储空间不同，为了充分利用存储空间，于是定义了不同的数据类型。而且，不同的数据类型，寓意也不同。

我们都知道，无论这个变量是字符串类型，还是数字类型，我们都可以直接用 `var` 去定义它。比如：

```
var a = 'hello word';

var b = 123;
```

为什么可以这样做呢？这是因为：JavaScript 是一种「弱类型语言」，或者说是一种「动态语言」，这意味着不需要提前声明变量的类型，在程序运行过程中，类型会自动被确定。

JS 的变量数据类型，是在程序运行的过程中，根据等号右边的值来确定的。而且，变量的数据类型是可以变化的。比如说：

```
var name = 'qianguyihao';

name = 123; // 强制将变量 name 修改为 数字类型
```

JS中一共有六种数据类型

- **基本数据类型（值类型）**：String 字符串、Number 数值、Boolean 布尔值、Null 空值、Undefined 未定义。
- **引用数据类型（引用类型）**：Object 对象。

注意：内置对象 Function、Array、Date、RegExp、Error等都是属于 Object 类型。也就是说，除了那五种基本数据类型之外，其他的，都称之为 Object类型。

> 面试问：引用数据类型有几种？

> 面试答：只有一种，即 Object 类型。

数据类型之间最大的区别：

- 基本数据类型：参数赋值的时候，传数值。
- 引用数据类型：参数赋值的时候，传地址（修改的同一片内存空间）。

今天这篇文章，我们详细讲一下基本数据类型。

String 字符串

语法

字符串型可以是引号中的任意文本，其语法为：双引号 `""` 或者单引号 `''`。

来看个示例。下面的这些，都是字符串：

```
    var a = "abcde";
    var b = "老胡";
```

```
    var c = "123123";
    var d = '哈哈哈哈哈';
    var e = "";     //空字符串

    var f = haha; // 没使用引号，到这里会直接报错。因为会被认为是js代码，但是之前并没有定义 haha。

    console.log(typeof a);
    console.log(typeof b);
    console.log(typeof c);
    console.log(typeof d);
    console.log(typeof e);

```

控制台输出如下：

```
    string
    string
    string
    string
    string
```

引号的注意事项

1、单引号和双引号不能混用。比如下面这样写是不可以的：

```
var str = 'hello";  // 报错：Uncaught SyntaxError: Invalid or unexpected token
```

2、同类引号不能嵌套：双引号里不能再放双引号，单引号里不能再放单引号。

3、单引号里可以嵌套双引号；双引号里可以嵌套单引号。

转义字符

在字符串中我们可以使用`\`作为转义字符，当表示一些特殊符号时可以使用`\`进行转义。

- `\"` 表示 `"`
- `\'` 表示 `'`
- `\\` 表示 `\`
- `\r` 表示回车
- `\n` 表示换行。n 的意思是 newline。
- `\t` 表示缩进。t 的意思是 tab。
- `\b` 表示空格。b 的意思是 blank。

举例：

```
    var str1 = "我说:\"今天\t天气真不错！\"";
    var str2 = "\\\\\\";

    console.log(str1);
    console.log(str2);
```

上方代码的打印结果：

```
    我说:"今天  天气真不错！"
    \\\
```

获取字符串的长度

字符串是由若干个字符组成的，这些字符的数量就是字符串的长度。我们可以通过字符串的 length 属性可以获取整个字符串的长度。

代码举例：

```
    var str1 = '老胡';
    var str2 = '老胡,前端.';

    var str3 = 'laohu';
    var str4 = 'laohu,qianduan.';

    console.log(str1.length); // 2
    console.log(str2.length); // 6
    console.log(str3.length); // 5
    console.log(str4.length); // 15
```

由此可见，字符串的 length 属性，在判断字符串的长度时，会认为：

- 一个中文算一个字符，一个英文算一个字符
- 一个标点符号（包括中文标点、英文标点）算一个字符
- 一个空格算一个字符

字符串拼接

多个字符串之间可以使用加号 `+` 进行拼接。

拼接语法：

```
字符串 + 任意数据类型 = 拼接之后的新字符串;
```

拼接规则：拼接前，会把与字符串相加的这个数据类型转成字符串，然后再拼接成一个新的字符串。

代码举例：（字符串与六大数据类型相加）

```
var str1 = '老胡前端' + '永不止步';
var str2 = '老胡前端' + 666;
var str3 = '老胡前端' + true;
var str4 = '老胡前端' + null;
var str5 = '老胡前端' + undefined;

var obj = { name: '老胡前端', age: 28 };
var str6 = '老胡前端' + obj;

console.log(str1);
console.log(str2);
console.log(str3);
console.log(str4);
console.log(str5);
console.log(str6);
```

打印结果：

```
老胡前端永不止步

老胡前端666

老胡前端true

老胡前端null

老胡前端undefined

老胡前端[object Object]
```

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 a);
```

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。无论整浮、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。

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。

另外，关于 isNaN() 函数，可以看下一篇文章。

浮点数的运算

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。

我们知道，所有的运算都要转换成二进制去计算，然而，二进制是无法精确表示1/10的。因此存在小数的计算不精确的问题。

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

输出：

```
我爱你
我+爱+你
6
```

总结：如果加号两边**都是**数值，此时是加。否则，就是连字符（用来连接字符串）。

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。

布尔值：Boolean

布尔型有两个值：true 和 false。主要用来做逻辑判断： true 表示真，false 表示假。

布尔值直接使用就可以了，千万不要加上引号。

代码：

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

控制台输出结果：

```
boolean
```

布尔型和数字型相加时， true 按 1 来算 ，false 按 0 来算。

Null 和 Undefined

`null`：空值

专门用来表示一个为空的**对象**（例如：`var a = null`）。注意，专门用来表示**空对象**。

- Null类型的值只有一个，就是null。比如`var a = null`。
- 使用 typeof 检查一个null值时，会返回object。

`undefined`：未定义

声明了一个变量，但是没有**赋值**（例如：`var a;`），此时它的值就是`undefined`。

- Undefined类型的值只有一个，就是undefind。比如
- 使用 type of 检查一个undefined时，会返回undefined。

null和undefined有很大的相似性。看看null == undefined的结果(true)也就更加能说明这点。

但是null === undefined的结果(false)。它们虽然相似，但还是有区别的，其中一个区别是：和数字运算时，10 + null结果为：10；10 + undefined结果为：NaN。

- 任何数据类型和undefined运算都是NaN;
- 任何值和null运算，null可看做0运算。

变量值的传递（赋值）

语句：

```
a = b;
```

把b的值赋给a，b不变。

将等号右边的值，赋给左边的变量；等号右边的变量，值不变。

来做几个题目。

举例1：

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

输出：

```
5
-2
-10
```

举例2：

```
                        //a    b     c
var a = 1;
var b = 2;
var c = 3;              //1     2     3
a = a + b;              //3     2     3
b = b + a;              //3     5     3
c = c + b;              //3     5     8
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
```