JavaScript 之数据类型

JavaScript 一共有 6 中数据类型：
基本数据类型（5）：字符串（String）、数字(Number)、布尔(Boolean)、数组(Array)、空（Null）、未定义（Undefined）
复杂数据类型（1）：对象(Object)
注意：Array、Date、Math、Error Set(ES6).....都是属于 Object 中

一、JS 数据类型概述

1.1 简介

原始类型（基本类型）：按值访问，可以操作保存在变量中实际的值。
原始类型汇总中 null、undefined 比较特殊。
引用类型：引用类型的值是保存在内存中的对象。

与其他语言不同的是，JavaScript 不允许直接访问内存中的位置，也就是说不能直接操作对象的内存空间。 在操作对象时，实际上是在操作对象的引用而不是实际的对象。所以引用类型的值是按引用访问的。

1.2 typeof 操作符

由于 js 中的变量是松散类型的，所以它提供了一种检测当前变量的数据类型的方法，也就是 typeof 关键字。

关键字	值	类型
typeof	123	Number
typeof	'abc'	String

关键字	值	类型
typeof	true	Boolean
typeof	undefined	Undefined
typeof	null	Object
typeof	{ }	Object
typeof	[ ]	Object
typeof	console.log()	Function

null 类型进行 typeof 操作符后，结果是 object，原因在于，null 类型被当做一个空对象引用。

二、原始类型

2.1 Number 类型

- Number 类型包含整数和浮点数（浮点数数值必须包含一个小数点，且小数点后面至少有一位数字）两种值。
- js 不区分 整型和 浮点型

  特殊值 NaN （非数字类型）

- NaN 跟任何值进行任何运算，结果仍然 NaN.跟谁都不相等，包括自己。
- 特点
  ① 涉及到的 任何关于 NaN 的操作，都会返回 NaN

  ② NaN 不等于自身。

- isNaN() 函数用于检查其参数是否是非数字值。
  isNaN(123) //false isNaN("hello") //true

一般 NaN 被动产生（数据类型转为 Number，不能转为正常的数字，就是 NaN）
函数 isNaN() 判断是不是 NaN 或者能不能转换为 NaN

使用实例：

```
        //数字
        var n1 = 10234;
        var n2 = 0x12; //十六进制
        var n3 = 2e2; //科学计数法（小学知识）

        console.log(n1,n2,n3)

        //浮点精度问题
        console.log(.1 + .2);

        //NaN  表示 Not a number

        console.log(NaN)
        console.log(typeof(NaN))      //NaN 的数据类型依然是 number

        //NaN 跟 任何值(包括 0) 进行任何运算 结果依然是 NaN
        console.log(NaN * 0);         //结果是 NaN

        //NaN 跟谁都不相等
        console.log(NaN == NaN)      //结果是 false


        var num = 2344e1000;

        console.log(typeof(num))     //结果是 infinity 即无穷大
        console.log(num)            //数据类型依然为 number
```

```
        console.log(isNaN(NaN))       //true
        console.log(isNaN('hello'))  //true
        console.log(isNaN('123')) // false 字符串'123' 转为 number 的时候 是 123 不是
NaN
```

2.2 String 类型

- 字符串是存储字符（比如 "Bill Gates"）的变量。
- 字符串有 length 属性
- 字符串可以是引号中的任意文本。您可以使用单引号或双引号(没有区别)。
- 您可以在字符串中使用引号，只要不匹配包围字符串的引号即可。

```
var answer="It's alright";
var answer="He is called 'Johnny'";
var answer='He is called "Johnny"';
```

2.3 Boolean 类型

布尔（逻辑）只能有两个值：true 或 false。

```
var x=true;
var y=false;
```

2.4 Null 类型 和 Undefined 类型

1) Null 类型

被动产生

null 类型被看做空对象指针，前文说到 null 类型也是空的对象引用。

2)Undefined 类型
只有一个值，即 undefined 值。使用 var 声明了变量，但未给变量初始化值，那么这个
变量的值就是 undefined.

Undefined 这个值表示变量不含有值。
可以通过将变量的值设置为 null 来清空变量。

```
cars=null;
person=null;
```

二、三大引用类型

js 中对象是一组属性与方法的集合。这里就要说到引用类型了，引用类型是一种数据结构，
用于将数据和功能组织在一起。引用类型有时候也被称为对象定义，因为它们描述的是一类
对象所具有的属性和方法。

2.1 Object 类型

我们看到的大多数类型值都是 Object 类型的实例，创建 Object 实例的方式有两种：
1）第一种是使用 new 操作符后跟 Object 构造函数

```
var person = new Object();
person.name = "Micheal";
person.age = 24;
```

2）第二种方式是使用对象字面量表示法
对象由花括号分隔。在括号内部，对象的属性以名称和值对的形式（name：value）

来定义。
属性由逗号分隔

```
var person = {
   name : "Micheal",
   age : 24
};
```

寻址的两种方式：

```
name=person.name;
name=person["name"];
```

2.2 Array 类型

数组的每一项可以用来保存任何类型的数据，也就是说，可以用数组的第一个位置来
保存字符串，第二个位置保存数值，第三个位置保存对象....另外，数组的大小是可
以动态调整的。

创建数组的基本方式有两种:
1）第一种是使用 Array 构造函数

```
var colors = new Array("red","blue","yellow");
 var cars=new Array();
 cars[0]="Saab";
 cars[1]="Volvo";
 cars[2]="BMW";
```

2）第二种是使用数组字面量表示法

```
var colors = ["red","blue","yellow"];
```

2.3 Function 类型

每个函数都是 Function 类型的实例，而且都与其他引用类型一样具有属性和方法。
函数通常是使用函数声明语法定义的，如下所示

```
function sum(num1,num2){
   return num1 + num2;
};
```

这和使用函数表达式定义函数的方式相差无几

```
var sun = function (){
   return sum1 + sum2;
};
```

注意：
当您声明新变量时，可以使用关键词 "new" 来声明其类型：

```
var carname=new String;
var x=     new Number;
var y=     new Boolean;
var cars=   new Array;
var person= new Object;
```

JavaScript 变量均为对象。当您声明一个变量时，就创建了一个新的对象。
提示：JavaScript 具有隐含的全局概念，意味着你不声明的任何变量都会成为一个全局对象属性。