4. JavaScript数据类型

JavaScript中数据类型：

- 值类型(基本类型)：

  字符串（String）、数字(Number)、布尔(Boolean)、对空（Null）、未定义（Undefined）、Symbol。

  注意：Symbol 是 ES6 引入了一种新的原始数据类型，表示独一无二的值。

- 引用数据类型： 对象(Object)、数组(Array)、函数(Function)。

```
//我们使用typeof操作符获取基本数据类型
//Undefined 这个值表示变量不含有值
console.log(typeof a); //undefined 未定义的

var a = 10;
console.log(typeof a); //number
a = 3.14
console.log(typeof a); //number

a = 'zhangsan'
console.log(typeof a); //string

a = true
console.log(typeof a); //boolean

//可以通过将变量的值设置为 null 来清空变量
a = null
console.log(typeof a); //object
console.log(a); //null

a = [10,20,30];
//或 a = new Array(10,20,30);
console.log(typeof a);  //object
console.log(a instanceof Array); //true

a = function(){} //定义空白函数
console.log(typeof a); //function
console.log(a instanceof Function); //true


/*
//整数的进制的输出
console.log(10); //输出十进制数的值
console.log(0b10); //输出二进制数10的值
```

```
console.log(0o10); //输出八进制数10的值
console.log(0x10); //输出十六进制数10的值

//十进制转换其他进制
var  x = 110;
x.toString(2)//转为2进制
x.toString(8)//转为8进制
x.toString(16)//转为16进制
//其他进制转十进制
var x = "110"//这是一个二进制的字符串表示
parseInt(x, 2)//把这个字符串当做二进制，  转为十进制

var x = "70"//这是一个八进制的字符串表示
parseInt(x, 8)//把这个字符串当做八进制，  转为十进制

var x = "ff"//这是一个十六进制的字符串表示
parseInt(x, 16)//把这个字符串当做十六进制，  转为十进制
*/
```

- typeof 操作符获取一个变量的类型,返回结果如下：
  - undefined - 如果变量是 Undefined 类型的
  - boolean - 如果变量是 Boolean 类型的
  - number - 如果变量是 Number 类型的 (整数、浮点数)
  - string - 如果变量是 String 类型的 （采用""、 ''）
  - object - 如果变量是一种引用类型或 Null 类型的 如：`new Array()/ new String()...`
  - function -- 函数类型

```
typeof "zhangsan"             // 返回 string
typeof 3.14                   // 返回 number
typeof NaN                    // 返回 number
typeof true                   // 返回 boolean
typeof [10,20,30,40]          // 返回 object
typeof {name:'lisi', age:20}  // 返回 object
typeof new Date()             // 返回 object
typeof function(){}           // 返回 function
typeof myCar                  // 返回 undefined (如果 myCar 没有声明)
typeof null                   // 返回 object
```

- undefined 和 null 的区别
  - null 和 undefined 的值相等，但类型不等：

```
typeof undefined     // undefined
typeof null          // object
null === undefined   // false
null == undefined    // true
```

- object引用类型 引用类型通常叫做类（class），也就是说，遇到引用值，所处理的就是对象。Object 对象自身用处不大，不过在了解其他类之前，还是应该了解它。 因为 ECMAScript 中的 Object 对象与 Java 中的 java.lang.Object 相似， ECMAScript 中的所有对象都由这个对象继承而来，Object 对象中的所有属性 和方法都会出现在其他对象中，所以理解了 Object 对象，就可以更好地理解其他对象。
- 值类型理解：变量之间的互相赋值，是指开辟一块新的内存空间，将变量值赋给新变量保存到新开辟的内存里面；之后两个变量的值变动互不影响，例如：

```
var a = 10; //开辟一块内存空间保存变量a的值“10”;
var b = a; //给变量 b 开辟一块新的内存空间，将 a 的值 “10” 赋值一份保存到新的内存里;
//a 和 b 的值以后无论如何变化，都不会影响到对方的值;
```

- 引用类型理解：变量之间的互相赋值，只是指针的交换，而并非将对象（普通对象，函数对象，数组对象）复制一份给新的变量，对象依然还是只有一个，只是多了一个指引。

```
//需要开辟内存空间保存对象，变量 a 的值是一个地址，这个地址指向保存对象的空间;
var a = { x: 1, y: 2 };
var b = a; // 将a 的指引地址赋值给 b，而并非复制一给对象且新开一块内存空间来保存;
// 这个时候通过 a 来修改对象的属性，则通过 b 来查看属性时对象属性已经发生改变;
```

类型转换：

- JavaScript 变量可以转换为新变量或其他数据类型：
  - 通过使用 JavaScript 函数
  - 通过 JavaScript 自身自动转换

```
ECMAScript 中可用的 3 种强制类型转换如下：
  Boolean(value) - 把给定的值转换成 Boolean 型;
  Number(value) - 把给定的值转换成数字（可以是整数或浮点数）;
  String(value) - 把给定的值转换成字符串;

使用：Number（）、parseInt() 和parseFloat（） 做类型转换
  Number()强转一个数值(包含整数和浮点数)。
  *parseInt()强转整数,
  *parseFloat（）强转浮点数

函数isNaN()检测参数是否不是一个数字。 is not a number
```

- 参考示例：

```
//转换字串类型
String(100 + 23)  // 返回 "123"
String(true)      // 返回 "true"
String(new Date())// 返回 "Tue May 14 2019 11:06:28 GMT+0800 (中国标准时间)"
String([10,20])   // 返回 "10,20"
String(null)    // 返回 "null"
```

```
//转换数值类型
Number("3.14")    // 返回 3.14
Number("3.14abc") // 返回 NaN
parseFloat("3.14")//返回 3.14
parseFloat("3.14abc")//返回 3.14
parseFloat("b3.14abc")//返回 NaN
parseInt("3.14")  //返回 3
parseInt("3.14abc")//返回 3
parseInt("b3.14abc")//返回 NaN
```

- 常见类型转换：

原始值	转换为数字	转换为字符串	转换为布尔值
false	0	"false"	false
true	1	"true"	true
0或"0"	0	"0"	false
1或 "1"	1	"1"	true
"000"	0	"000"	true
NaN	NaN	"NaN"	false
""	0	""	false
"字串"	NaN	"字串"	true
[ ]	0	""	true
[10,20,30]	NaN	"10,20,30"	true
function(){}	NaN	"function(){}"	true
{ }	NaN	"[object Object]"	true
null	0	"null"	false
undefined	NaN	"undefined"	false