# JS数据类型（基本数据类型+引用类型）

数据类型指的是可以在程序中存储和操作的值的类型，每种编程语言都有其支持的数据类型，不同的数据类型用来存储不同的数据，例如文本、数值、图像等。

JavaScript 是一种动态类型的语言，在定义变量时不需要提前指定变量的类型，变量的类型是在程序运行过程中由 JavaScript 引擎动态决定的，另外，您可以使用同一个变量来存储不同类型的数据，例如：

```
var a;  // 此时 a 为 Undefined
a = "http://c.biancheng.net/"; // 此时 a 为 String 类型
a = 123;  // 此时 a 为 Number 类型
```

JavaScript 中的数据类型可以分为两种类型：

- 基本数据类型（值类型）：字符串（String）、数字（Number）、布尔（Boolean）、空（Null）、未定义（Undefined）、Symbol；
- 引用数据类型：对象（Object）、数组（Array）、函数（Function）。

> 提示：Symbol 是 ECMAScript6 中引入的一种新的数据类型，表示独一无二的值。

**typeof 操作符**

在开始介绍各种数据类型之前，先来了解一下 typeof 操作符，使用 typeof 操作符可以返回变量的数据类型。

typeof 操作符有带括号和不带括号两种用法，如下例所示：

```
typeof x;       // 获取变量 x 的数据类型
typeof(x);      // 获取变量 x 的数据类型
```

## 1. JS 基本数据类型

**1) String 类型**

字符串（String）类型是一段以单引号 `''` 或双引号 `""` 包裹起来的文本，例如 '123'、"abc"。需要注意的是，单引号和双引号是定义字符串的不同方式，并不是字符串的一部分。

定义字符串时，如果字符串中包含引号，可以使用反斜杠 `\` 来转义字符串中的引号，或者选择与字符串中不同的引号来定义字符串，如下例所示：

```
var str = "Let's have a cup of coffee.";  // 双引号中包含单引号
var str = 'He said "Hello" and left.';    // 单引号中包含双引号
var str = 'We\'ll never give up.';        // 使用反斜杠转义字符串中的单引号
```

**2) Number 类型**

数值（Number）类型用来定义数值，JavaScript 中不区分整数和小数（浮点数），统一使用 Number 类型表示，如下例所示：

```
01.  var num1 = 123;      // 整数
02.  var num2 = 3.14;     // 浮点数
```

> 注意：Number 类型所能定义的数值并不是无限的，JavaScript 中的 Number 类型只能表示 -($2^{53}$ - 1) 到 ($2^{53}$ -1) 之间的数值。

对于一些极大或者极小的数，也可以通过科学（指数）计数法来表示，如下例所示：

```
01.  var y=123e5;       // 123 乘以 10 的 5 次方，即 12300000
02.  var z=123e-5;      // 123 乘以 10 的 -5 次方，即 0.00123
```

另外，Number 类型中还有一些比较特殊的值，分别为 Infinity、-Infinity 和 NaN，其中

- Infinity：用来表示正无穷大的数值，一般指大于 1.7976931348623157e+308 的数；
- -Infinity：用来表示负无穷大的数值，一般指小于 5e-324 的数；
- NaN：即非数值（Not a Number 的缩写），用来表示无效或未定义的数学运算结构，例如 0 除以 0。

> 提示：如果某次计算的结果超出了 JavaScript 中 Number 类型的取值范围，那么这个数就会自动转化为无穷大，正数为 Infinity，负数为 -Infinity。

**3) Boolean 类型**

布尔（Boolean）类型只有两个值，true（真）或者 false（假），在做条件判断时使用的比较多，您除了可以直接使用 true 或 false 来定义布尔类型的变量外，还可以通过一些表达式来得到布尔类型的值，例如：

```
01.  var a = true;   // 定义一个布尔值 true
02.  var b = false;  // 定义一个布尔值 false
03.  var c = 2 > 1;  // 表达式 2 > 1 成立，其结果为“真（true）”，所以 c 的值为布尔类型的
     true
04.  var d = 2 < 1;  // 表达式 2 < 1 不成立，其结果为“假（false）”，所以 c 的值为布尔类
     型的 false
```

**4) Null 类型**

Null 是一个只有一个值的特殊数据类型，表示一个“空”值，即不存在任何值，什么都没有，用来定义空对象指针。

使用 typeof 操作符来查看 Null 的类型，会发现 Null 的类型为 Object，说明 Null 其实使用属于 Object（对象）的一个特殊值。因此通过将变量赋值为 Null 我们可以创建一个空的对象。

**5) Undefined 类型**

Undefined 也是一个只有一个值的特殊数据类型，表示未定义。当我们声明一个变量但未给变量赋值时，这个变量的默认值就是 Undefined。例如：

```
var num;
console.log(num);  // 输出 undefined
```

在使用 typeof 操作符查看未赋值的变量类型时，会发现它们的类型也是 undefined。对于未声明的变量，使用 typeof 操作符查看其类型会发现，未声明的变量也是 undefined，示例代码如下：

```
var message;
console.log(typeof message);  // 输出 undefined
console.log(typeof name);     // 输出 undefined
```

#### 6) Symbol 类型

Symbol 是 ECMAScript6 中引入的一种新的数据类型，表示独一无二的值，Symbol 类型的值需要使用 Symbol() 函数来生成，如下例所示：

```
var str = "123";
var sym1 = Symbol(str);
var sym2 = Symbol(str);
console.log(sym1);           // 输出 Symbol(123)
console.log(sym2);           // 输出 Symbol(123)
console.log(sym1 == sym2);   // 输出 false ：虽然 sym1 与 sym2 看起来是相同的，但实际上它们并不一样，根据 Symbol 类型的特点，sym1 和 sym2 都是独一无二的
```

## 2. JS 引用数据类型

#### 1) Object 类型

JavaScript 中的对象（Object）类型是一组由键、值组成的无序集合，定义对象类型需要使用花括号 `{}`，语法格式如下：

```
{name1: value1, name2: value2, name3: value3, ..., nameN: valueN}
```

其中 name1、name2、name3、...、nameN 为对象中的键，value1、value2、value3、...、valueN 为对应的值。

在 JavaScript 中，对象类型的键都是字符串类型的，值则可以是任意数据类型。要获取对象中的某个值，可以使用 `对象名.键` 的形式，如下例所示：

```
var person = {
    name: 'Bob',
    age: 20,
    tags: ['js', 'web', 'mobile'],
    city: 'Beijing',
    hasCar: true,
    zipcode: null
```

```
};
console.log(person.name);        // 输出 Bob
console.log(person.age);         // 输出 20
```

**2) Array 类型**

数组（Array）是一组按顺序排列的数据的集合，数组中的每个值都称为元素，而且数组中可以包含任意类型的数据。在 JavaScript 中定义数组需要使用方括号 `[]`，数组中的每个元素使用逗号进行分隔，例如：

```
[1, 2, 3, 'hello', true, null]
```

另外，也可以使用 Array() 函数来创建数组，如下例所示：

```
var arr = new Array(1, 2, 3, 4);
console.log(arr);        // 输出 [1, 2, 3, 4]
```

数组中的元素可以通过索引来访问。数组中的索引从 0 开始，并依次递增，也就是说数组第一个元素的索引为 0，第二个元素的索引为 1，第三个元素的索引为 2，以此类推。如下例所示：

```
var arr = [1, 2, 3.14, 'Hello', null, true];
console.log(arr[0]);  // 输出索引为 0 的元素，即 1
console.log(arr[5]);  // 输出索引为 5 的元素，即 true
console.log(arr[6]);  // 索引超出了范围，返回 undefined
```

**3) Function 类型**

函数（Function）是一段具有特定功能的代码块，函数并不会自动运行，需要通过函数名调用才能运行，如下例所示：

```
function sayHello(name){
    return "Hello, " + name;
}
var res = sayHello("Peter");
console.log(res);  // 输出 Hello, Peter
```

此外，函数还可以存储在变量、对象、数组中，而且函数还可以作为参数传递给其它函数，或则从其它函数返回，如下例所示：

```
var fun = function(){
    console.log("http://c.biancheng.net/js/");
}

function createGreeting(name){
    return "Hello, " + name;
}
function displayGreeting(greetingFunction, userName){
    return greetingFunction(userName);
}
var result = displayGreeting(createGreeting, "Peter");
```

```
12.  console.log(result);  // 输出 Hello, Peter
```