

ECMAScript基础

JS分为三部分，其中 **ECMAScript** 规定了这门语言的语法基础，数据类型，流程控制，运行符，类型转换，函数，面向对象等一系列的基础知识，它是入门点

ECMAScript的语法基础规则

```
1  <!DOCTYPE html>
2  <html lang="zh">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>基础规则</title>
8      <STYLE>
9          INPUT{
10              BACKGROUND-COLOR:RED;
11          }
12      </STYLE>
13 </head>
14 <body>
15     <input type="text">
16     <INPUT TYPE="TEXT" />
17 </body>
18 <script>
19     alert("陈怡静,刘诗霞,芦伊菲");
20     // ALERT("陈怡静,刘诗霞,芦伊菲");    这里不行
21 </script>
22 </html>
```

1. **JS** 代码的基础单位是行，一行代码结束以后要以分号结尾【如果省略了这个分号也是可以的，但是不推荐】
2. **JS** 是一门严格区分大小写的语言，它与之前的 **HTML**、**CSS** 不一样
3. **JS** 编程里面，所有的符号都是英文状态下面的符号

变量标识符

变量其实就是一个可以变化内容或值的一个符号，在之前的CSS里面，我们已经接触过了变量的概念

```
1  :root{
2      --primaryColor:red;
3  }
4  .box{
5      color:var(--primaryColor);
6  }
```

通过上面的过程，我们可以大概知道变量是，它定义的过程和使用的过程是什么样式

在JS里面也是一样的，在JS当中定义变量有一个特殊的**关键字** **var** 来实现的

`var` 的全名叫 `variable` , 英文单词就是变量的意思


```
1 var a;
```

上面的代码就是定义了变量 `a` 。这行代码的意思就是告诉浏览器, 我 `var` 后面跟着的这个 `a` 它是一个变量,但是这个变量没有赋值, 没有赋值的变量叫**未初始化的变量**

```
1 var a = 18;
2 //我们可以在定义变量的时候, 直接给这一个变量赋一个初始的值, 这个赋值始值的过程叫 变量初始化
```

上面的代码其实也可以分2部去写

```
1 var a; //定义变量
2 a = 18; //赋值
```

 通过上面的代码我们已经知道 关键字 `var` 就是用来定义变量的, 但是这里有2个细节需要同学样注意

1. 在 `JS` 里面, 定义所有的变量都使用同一个关键字 `var` , 这一点与其它的语言不一样

```
1 var a = 18;
2 var b = "标哥哥";
3 var c = 19.9;
4 var d = true;
```

`JS`它是一门**弱类型**的语言【弱类型就是在定义变量的时候, 不去强调变量的类型, 同时变量的类型也是可以后期进行相互转换的】

```
1 //下面是java的代码
2 int a = 18;
3 String name = "标哥哥";
4 char sex = '男';
5 float money = 11.2f;
```

2. 变量的命名一定要规范

- 所有的变量名应该遵守见名知意
- 所有的变量命名应该是驼峰命名法, 驼峰命名法就是第一个单词首字母小写, 后面单词的首字母大写
- 所有的变量命名应该避开关键字与保留字
- 变量只能是字母, 下划线或 `$` 开头

```
1 //定义年龄使用age 见名知意
2 var age = 18;
3 //定义姓名使用username 见名知意
4 var userName = "标哥";
5 // 学生信息列表的变量
6 var studentInfoList;
7
8 // var var; 错误的变量命名 使用了关键字做变量名
9 // var for; 错误的变量命名 使用了关键字做变量名
10
11 //var 1age; 错误的变量命名 使用了数字开头
12 // var user-name; 错误的变量命名 使用了特殊字符做变量名
13 // var #age; 错误的变量命名 使用了特殊字符做变量名
14
```

```
15  var $age = 20;
16  var user_age = 20;
17  var _age = 20;
```

关键字

关键字是编译系统里面为了实现一个或多个功能所特定命名的一个关键词

ECMA-262 描述了一组具有特定用途的**关键字**，这些关键字可用于表示控制语句的开始或结束，或者用于执行特定操作等。按照规则，关键字也是语言保留的，不能用作标识符。以下就是 ECMAScript 的全部关键字（带*号上标的是第 5 版新增的关键字）：

break	do	instanceof	typeof
case	else	new	var
catch	finally	return	void
continue	for	switch	while
debugger*	function	this	with
default	if	throw	
delete	in	try	

保留字

保留字也是关键字，它在当前的版本中没有使用，但是可能会有后面的版本里面去使用

ECMA-262 还描述了另外一组不能用作标识符的**保留字**。尽管保留字在这门语言中还没有任何特定的用途，但它们有可能在将来被用作关键字。以下是 ECMA-262 第 3 版定义的全部保留字：

abstract	enum	int	short
boolean	export	interface	static
byte	extends	long	super
char	final	native	synchronized
class	float	package	throws
const	goto	private	transient
debugger	implements	protected	volatile
double	import	public	

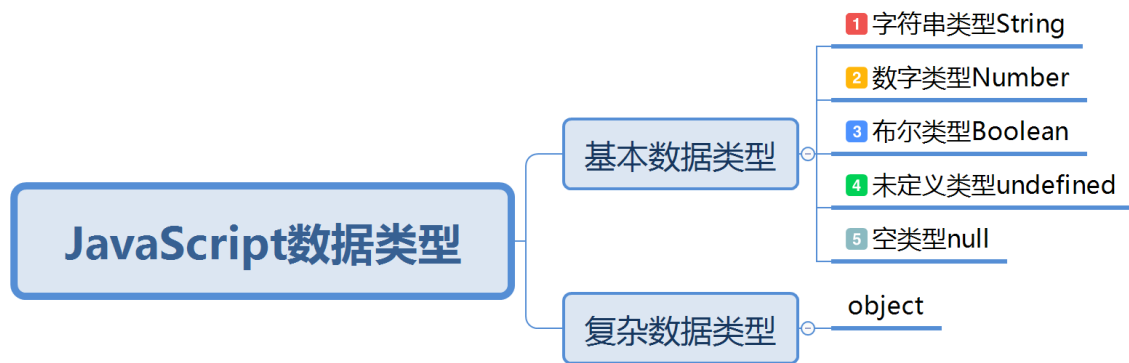
现在我们已经知道变量的定义是使用关键字 **var** 了，那么们就来通过这个关键来带出新的知识点

```
1  var teacherName = "标哥";
2  var age = 18;
3  var height = 180.5;
4  var money = null;
```

我们可以看到变量后面跟着的这些些是不一样的，要弄清楚上面的问题，我们就需要学习 **ECMAScript** 当中的数据类型

JavaScript数据类型

在JavaScript里面根据不同的场景需求，我们要使用一些特殊的数据来去描述，如数字，如逻辑判断的真或假，如小数等，这些都称之为数据类型



在JS的数据类型里面，我们把数据类型分为2个大类

1. 基本数据类型

基本数据类型描述数据的根本类型，在JS里面一定分为很多小的类

2. 复杂数据类型【引用类型】

字符串类型String

字符串是所有编程语言当中最常见的数据类型，同时在JS里面也是最安全的数据类型，所有数据类型都可以转换成这种类型，它也非常好识别（所有使用引号包裹的都是字符串）

```
1 var a = "标哥哥";
2 var b = "hello";
3 var c = "100";
```

上面变量赋的值全都被引号包裹了，但是要注意，符号全都是英文的

```
1 var d = "标哥哥";
```

上面的代码就是错误的，它使用了中语文的符号，编程语言里面所有的符号应该都是英文的

在字符串类型里面所有被引号包裹的就是字符串，没有单引号与双引号的区别，**只要是成双成对的出现就可以了**

```
1 var sex = '男'; //正确
2 var stuName = '黄相立'; //正确
3 var girlFriend = '赵金麦'; //这是不对的，因为引号不是成对的出现的
```

现在请同学样来判断一下，下面是不是字符串

```
1 var a = 18; //数字
2 var b = "18"; //字符串
3 var c = '18'; //字符串
4 var d = c; //相当于 var d = '18', 它也是字符串
5 var e = 'c';
6 //第一个a没有打引号，所以不是字符串
7 //最后的d是接收了c的值，而c的值是'18', 所以变量d最后的值也是'18'
```

数字类型Number

数字类型是JavaScript当中用于表述数字的，最简单的数字类型就是平时使用的10进制数据类型（后面我们还会讲到8进制，16进制及2进制）

后面会讲到2进制及进制转换，所以这里只写10进制

同时JS里面的数字类型与其它的语言不一样，它不区分小数，整数，长整型，短整型，单精度，双精度等，所有的数字全称之为 **Number** 类型

```
1  var money = 0;           //整数
2  var money1 = 25;         //数字
3  var money2 = 35.5;       //小数
4  var money3 = -100;       //负数
5  var a = "999";          //只要加了引号，就是字符串类型
```

现在再列举一下16进制的情况【了解】，所以的16进制的数字都是以 **0x** 开头的16进制方式来赋值

```
1  var a = 255;             //十进制
2  var b = 0xff;            //0x一定是16进制 后面的ff代表提16进制的值
```

16进制的值一旦赋值结束以后，系统会自动的转换成10进制的值

所有的十进制数字都是由0~9构成的，但是仍然有一些特殊的数字要记住

关于数字类型NaN

NaN 是一个非常特殊的数字类型，全称叫 **not a number**，这个值代表本来经过运算以后应该得到一个数字的，结果又得不到数字出现了意外，就会显示这个值

```
1  var a = 100;
2  var b = "标哥";
3  a - b; //这个时候的结果就是NaN
4  //减法操作以后按结果应该是一个数字，结果我又得不到数字，出现了意外，这个时候就会得到NaN
```

布尔类型Boolean

布尔类型是绝大多编程语言里面都具备的数据类型，这个类型 下面只有2个值，分别是 **true** 和 **false**，用于在编程过程当当中判断条件真或假

```
1  var isTeacher = true;
2  var isGirl = false;
3  var x = 1;
4  var y = 2;
5  var z = (x < y); //这条件是成立的，所以它的结果是一个true
```

请同学们分析一下下面的数据类型是什么

```
1  var a = "true";
2  var b = false;
3  var c = 'false';
```

变量a被引号包含，所以它是字符串，变量b是 **false** 所以是Boolean类型，变量c被单引号包裹，也是字符串

未定义类型undefined

未定义类型是 **JavaScript** 当中独有的一种特殊类型，它指的是一个变量定义了以后没有初始化或在初始化的时候给了一个 **undefined** 的值

```
1 var a = true;           //boolean
2 var b = 123;            //number
3 var c = "hello";        //string
```

上面的3个变量都有数据类型，原因是因为它们都有值，但是如果一个变量定义了以后没有值，会是什么类型

```
1 var d ;    //这个时候就只定义了变量，而没有初始化，它就是一个未定义的类型 undefined
```

还有一种情况也可以是未定义类型，就是直接赋一个 **undefined** 的值

```
1 var e = "赵今麦";       //string类型
2 e = undefined;           //undefined类型
```

所以 **undefined** 代表没有的意思

空类型Null

null与我们上面所学习的 **undefined** 非常相似，它也代表没有，但是2个类型有本质性的区别

```
1 var a ;                //不赋值它就是undefined类型
2 var b = null;           //空类型
```

b变量null相当于一个房子，这个房子它是一个空房子,a是undefined，它代表空，它连房子都没有

场景一对话：undefined的情况对象

```
1 杨标：“颜一鸣，来，我今天晚上给一个大大的礼物给你”
2 颜一鸣听了以后非常高兴，跑跑跳跳就回到座位
3 李心悦上线了
4 问：“嘿，小伙子，刚刚标哥叫你过去干什么”
5 颜一鸣神秘秘的说道：“标哥说晚上给一个礼物我”
6 李心悦就问：“啥礼物，多少钱？”
7 颜一鸣听后说道：“我也不知道，标哥说晚上给我”
```

场景二对话：null 的情况的对话

```
1 杨标：“颜一鸣，来，小伙子，这里有个盒子，盒子里面装的是你今天晚上的惊喜礼物”
2 抱着这个盒子，颜一鸣欢乐的走向啊自己的位置
3 李心悦又上线了
4 “哟，不错啊，又有东西过来了，你和标哥啥关系，怎么老有礼物给你，快看看，到底是个啥？”
5 颜一鸣很好奇的就打开了这个盒子，然后发现，这个盒子是这空的
6 颜一鸣大失所望，然后就开始慢慢问候标哥了
```

上面的2个场景对话就充分的说明了 **null** 与 **undefined** 的场景，而给颜一鸣的盒子就相当于是在内存当中的空间，**null** 是会在内存里面占据一个空间，只是这个空间里面没有放任何东西，而 **undefined** 在内存里面是没有空间的

数据类型检测

JavaScript有5种基本数据类型，如果一个变量赋了值以后就有了初始的数据类型，那么这个数据类型是否会发生变化

```
1 var a = "hello";
2 a = "你好";
```

上面的代码里面，我们定义了一个变量，同时对这个变量进行了初始化，它的初始数据类型是 `String` 字符串类型，后面把 `a` 重新赋值为“你好”的时候它还是字符串类型

```
1 var a = "hello";    //字符串
2 a = 100;            //这里不会报错，现在赋值数字了，它就是number字符串类型
```

从上面的代码上面我们可以看到，`JavaScript` 的数据类型是可以发生改变的，这一种现象我们叫 `弱类型机制`

在 `JS` 当中变量是没有固定的数据类型的，它的数据类型是由后面的值来决定的，我们赋什么类型值，它就是什么数据类型

这一种现象与其它的编程语言是不一样的，`java` 与 `c` 等都是强类型机制，一旦确定了变量的数据类型，就不可能更改

```
1 int a;
2 double b;
3 float c;
4 String d;
5 boolean f;
```

正是因为JS的数据类型在更改，所以如果我们想确定一个变量的数据类型，可以通过 `数据类型检测` 来完成

在JS里面，基本数据类型的检测使用的是关键字 `typeof`

```
1 var a = 1;
2 var b = "你好";
3 var c = false;
4 var d = undefined;
5 var e = null;
6 // 现在一下一个检测
7 typeof a; // 'number' 类型
8 typeof b; // 'string' 类型
9 typeof c; // 'boolean' 类型
10 typeof d; // 'undefined' 类型
11 typeof e; // 'object' 类型
```

在前面的4个数据检测的时候，得到的结果与我们预期的结果是一致的，只有 `null` 在检测的时候得到的结果是 `object` 对象类型

基本数据类型检测以后得到的结果有以下几种情况

1. `number` 数字类型
2. `string` 字符串类型
3. `boolean` 布尔类型
4. `undefined` 未定义类型

5. `object` 对象类型，这个结果比较特殊，我们的 `null` 检测出来会是这个类型【单独对待】

数据类型检测的语法

```
1  typeof 变量名;
```

除了上面的方式以外，还可以使用下面的方式也行

```
1  typeof(变量名);
```

上面2种写法它的结果是一样的，只是写法不一样而已

顺带着说一句

```
1  typeof NaN;  //'number' 类型
2
3  var a = 123;
4  typeof typeof a;
5
6  //分析过程
7  //typeof(typeof(a));  //'string'
8  // typeof a  //'number'
9
10 //typeof 'number'  //结果'string'
```

数据类型转换

JavaScript里面有5种基本数据类型，这5种基本数据类型之间是可以发生相互的转换的

字符串转十进制数值

```
1  var a = "999";
2  var b = "9a9";
3  var c = "99.99";
4  var d = "a99";
5  var e = "99.99.88";
6  var f = "标哥";
7  var g = "-99";
8  var h = "";
```

这里我们定义了7个字符串的变量，那么如何将上面的这些字符串转换成相应的数字 `Number` 类型呢？

`String` 字符串转 `Number` 有三个方法

1. 通过 `Number(value?: any): number` 方法来完成转换

通过这种方式的转换，只能实现合法的数字字符串转换


```

1  var a1 = Number(a);      //得到结果999,它是一个number
2  var b1 = Number(b);      //得到结果NaN, 转换失败, 它也是一个number类型
3  var c1 = Number(c);      //得到结果99.99 它是一个number类型
4  var d1 = Number(d);      //得到结果NaN
5  var e1 = Number(e);      //得到结果NaN
6  var f1 = Number(f);      //NaN
7  var g1 = Number(g);      //得到结果-99 它是一个number类型
8  var h1 = Number(h);      //得么结果0

```

同时我们再看下面的几种情况

```

1  var h = "0001";
2  var h1 = Number(h);    //1
3
4  var aa = "0x0001";
5  var aa1 = Number(aa);  //1

```

`Number` 方法的转换本质上面就是直接去掉了字符串的引号, 然后再看它是否是一个合法的数字

后期所有的隐式类型转换全都默认使用这个方法进行转换

2. 使用 `parseInt(value: string, radix?: number): number` 进行转换

这个方法只能将字符串转换成整数

```

1  var a1 = parseInt(a);    //999
2  var b1 = parseInt(b);    //9 尝试进行转换, 成功就保留, 失败了就不要了
3  var c1 = parseInt(c);    //99
4  var d1 = parseInt(d);    //NaN
5  var e1 = parseInt(e);    //99
6  var f1 = parseInt(f);    //NaN
7  var g1 = parseInt(g);    //-99

```

`parseInt` 的转换是逐个转换, 成功就保留, 失败就停止, 如果一开始就失败了则返回 `NaN`。同时要注意, 它的结果成功以后也只会是一个整数

3. 使用 `parseFloat(value:string):number` 进行转换

这个结果与上面的结果非常相似, 只是会保留小数点以后的内容

```

1  var a1 = parseFloat(a);  //999
2  var b1 = parseFloat(b);  //9
3  var c1 = parseFloat(c);  //99.99
4  var d1 = parseFloat(d);  //NaN
5  var e1 = parseFloat(e);  //99.99
6  var f1 = parseFloat(f);  //NaN
7  var g1 = parseFloat(g);  //-99
8
9  //特殊情况
10 var x = parseFloat("99.0"); //99
11 var y = parseFloat(".9");   //0.9

```

布尔类型转数字

这里指的是布尔类型转10进制数字

布尔类型如果想转 `number` 类型，只能通过 `Number(value?: any): number` 来进行

```
1  var a = true;
2  var b = false;
3
4  //如何将上面的2个值转换成Number类型
5
6  var a1 = Number(a); //结果1
7  var b1 = Number(b); //结果0
```

❓ 思考：为什么这里不使用 `parseInt/parseFloat` 去转换

null与undefined转换为数值

```
1  var a = null;
2  var b = undefined;
3  // 想转数字，到底什么方法？
4  // Number(value?: any): number
5  // parseInt(value: string, radix?: number): number
6  // parseFloat(value:string):number
```

通过分析以后，我们得到应该是要通过 `Number` 去转换

```
1  var a1 = Number(null);          //0
2  var b1 = Number(undefined);     //NaN
```

其它类型转字符串

字符串是JS里面最根本的数据类型，所以的数据类型都可以转换为字符串

```
1  var a = 1;
2  var b = false;
3  var c = null;
4  var d = undefined;
5  var e = 3.14;
6  var f = -10;
7  var g = NaN;
```

上面的所有变量都是非字符串类型的，那么我们现在要将这些数据转换成字符串，怎么办呢？

字符串的转换非常简单，并且方法还非常多

1. 使用 `String(value?:any):string` 方法来进行转换

```
1  var a1 = String(a);           //"1";
2  var b1 = String(b);           //"false";
3  var c1 = String(c);           //"null";
4  var d1 = String(d);           //"undefined";
5  var e1 = String(e);           //"3.14";
6  var f1 = String(f);           //"-10";
7  var g1 = String(g);           //"NaN";
```

2. 直接将这个变量加上一个空字符串

```

1 // 直接加空字符串
2 var a1 = a + "";           //"1";
3 var b1 = b + "";           //"false";
4 var c1 = c + "";           //"null";
5 var d1 = d + "";           //"undefined";
6 var e1 = e + "";           //"3.14";
7 var f1 = f + "";           //"-10";
8 var g1 = g + "";           //"NaN";

```

这个特性非常重要，在JS里面，执行的是字符串优先

3. 通过 `toString()` 的方法来完成，这一种方法只适用于非 `null` 与非 `undefined` 的数据类型

```

1 // 通过toString来转换
2 var a1 = a.toString();      //"1";
3 var b1 = b.toString();      //"false";
4 // var c1 = c.toString();    //直接报错,因为null不具备 toString()的方法
5 // var d1 = d.toString();    //直接报错,因为undefined不具备 toString()的方法
6 var e1 = e.toString();      //"3.14";
7 var f1 = f.toString();      //"-10";
8 var g1 = g.toString();      //"NaN";

```

字符串的转换有3种方法，经过对比我们发现前2种方法的转换是没有任何限制条件的，后面的 `toString` 方法则不能在 `null` 和 `undefined` 的下面使用

其它类型转布尔值

重点中的重点

在JS所有的数据类型里面，只有6个明确的 `false` 条件，它分别是 空字符串 `""`，`0`, `false`, `null`, `undefined`, `NaN`

这一个转换过程非常麻烦，因为如果布尔类型的结果不明确，那么我们后面在学习流程控制的时候就会有问题，如 `if...else`, `for`, `while`, `do...while`

在JS里面，如果要将其它类型的值转换成布尔类型，则优先使用 `Boolean(value?:any):boolean` 这一个方法来进行

```

1 var a = 1;
2 var b = 0;
3 var c = "true";
4 var d = "false";
5 var e = "";
6 var f = null;
7 var g = undefined;
8 var h = NaN;
9 var i = 123;
10 var j = "标哥哥";
11
12 // 上面的所有值都要通过Boolean去转换

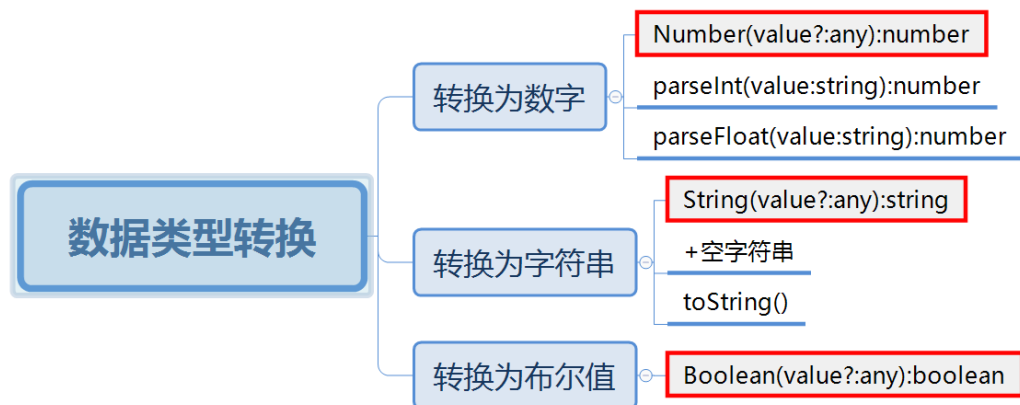
```

在上面的变里里面，我们把所有的数据类型基本上都涵盖了，现在将它们转换成布尔类型，看结果是什么

```

1  var a1 = Boolean(a);           //true
2  var b1 = Boolean(b);           //false
3  var c1 = Boolean(c);           //true
4  var d1 = Boolean(d);           //true
5  var e1 = Boolean(e);           //false
6  var f1 = Boolean(f);           //false
7  var g1 = Boolean(g);           //false
8  var h1 = Boolean(h);           //false
9  var i1 = Boolean(i);           //true
10 var j1 = Boolean(j);           //true

```



所有打了红色标记的，都是后面隐式数据类型转换所使用的默认方式

数字进制之间的转换

之前我们在学习 `Number` 数据类型的时候，我们只是对10进制的数值做了学习，其它数字还可以转换成其它的进制，也可以将其它的进制转换成10进制，现在我们来看看它如何操作

十进制转其它进制

十进制转其它进制，只能转到2~36

```

1  var a = 37;
2  var b = 18;

```

上在的2个数就是10进制的数，如何将这2个数转换成2进制呢

在大学里面学习的时候，老师都教过我们可以使用**除2取余法**来进行

$$\begin{array}{r}
 2 \overline{) 371} \\
 \underline{218} \\
 2 \overline{) 91} \\
 \underline{18} \\
 2 \overline{) 40} \\
 \underline{40} \\
 2 \overline{) 20} \\
 \underline{20} \\
 1
 \end{array}$$

1 0 0 1 0 1

上面的结果是手动算出来的，但是我们程序员也可以通过程序来计算

在数字类型 `Number` 里面，有一个方法叫 `toString(radix?:number)`，在数字类型转进制的时候，我们需要调用这个方法，其中 `radix` 代表进制基数,也叫目标进制

```

1  var a1 = a.toString(2);           //二进制的'100101'
2  var b1 = b.toString(2);           //二进制的'10010'
3  var c1 = c.toString(2);           //二进制的'1000001'

```

所以10进制转换成其它进制是非常简单的只需要将调用 `toString(目标进制)` 就可以了，如果不写目标进制，则默认转换成10进制

```

1  toString(目标进制);    //不填目标进制就是10进制

```

这个方法不仅可以转换成二进制，还可以转换成任何其它的进制都可以

```

1  var d = 255;
2  d.toString(16);           //十六进制的"ff"

```

其它进制转十进制

如果原来的进制数是0，则会直接忽略

```

1  var a = "10010";           //这是2进制的10010
2  var b = "1001001";         //这是2进制的1001001

```

如何将上面的2个数转换成10进制呢

在大学里面，我们也学过了一个手动转换的规则叫**指数相加法**

1	0	0	1	0
---	---	---	---	---

$$(2^4 \times 1) + (2^3 \times 0) + (2^2 \times 0) + (2^1 \times 1) + (2^0 \times 0)$$

$$16 + 0 + 0 + 2 + 0$$

上面的图就是一个手动计算的过程，但是我们也可以通过程序来计算

其实转10进制的方法我们也学习过了，它主要是使用

`parseInt(value:string,radix:number):number` 来实现的，虽然说我们之前已经学习过了这个方法，但是我们并没有使用第二个参数，这个方法后面的第二个值代表这个要转换的数原来的进制

```
1 parseInt(原来的值,原位的进制); //如果不填，则默认就是10
```

现在有了上面的代码基础以后，我们可以试着来进行相关的操作

```
1 var a1 = parseInt(a,2); //18
2 var b1 = parseInt(b,2); //73
```

现在请同学们看一下下面这个代码会是什么结果

```
1 var c1 = parseInt("ff"); //转换失败，得到结果NaN
2 var d1 = parseInt("ff",16); //转换成功，得到结果255
3 var e1 = parseInt("3",2); //转换失败，得到结果NaN
4 var f1 = parseInt("3",3); //转换失败，得到结果NaN
5 var g1 = parseInt("3",0); //转换成功，忽略0进制，直接使用10进制，结果是3
```

语法格式

```
0 var i = 你;
7 var g = "-99";
```

这里我们定义了7个字符串的变量，那么如何将上面的这些字符串转换成相应的数字 `Number` 类型呢？

`String` 字符串转 `Number` 有三个方法

1. 通过 `Number(value?: any): number` 方法来完成转换

可以有，也可以没有

放进去的值

结果是什么类型

放进去值的类型