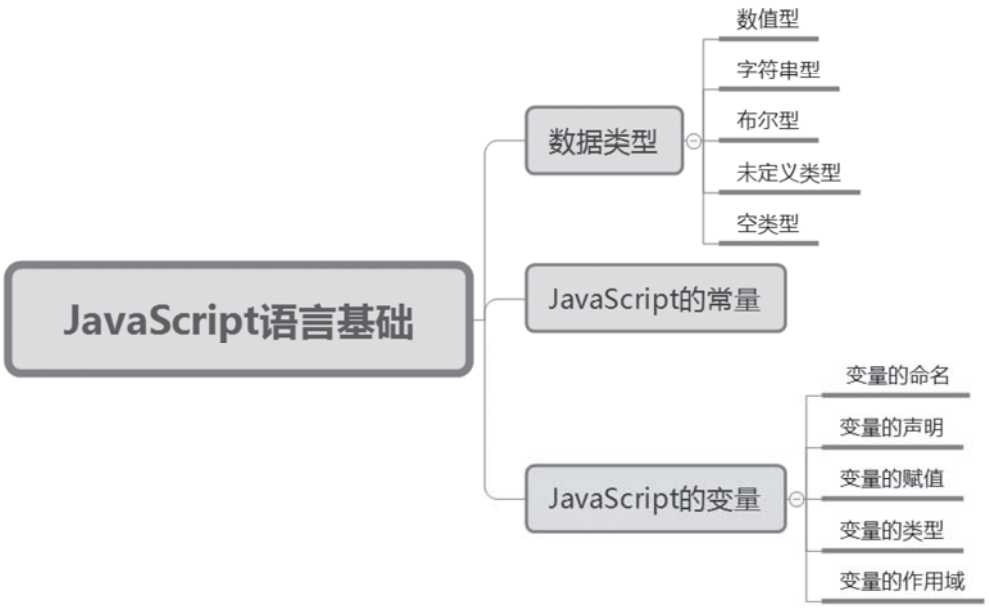
第2章　JavaScript语言基础

本章导读

无论是传统编程语言，还是脚本语言，都有自己的语言基础，JavaScript 脚本语言也不例外。本章就来介绍 JavaScript 的语言基础，包括数据类型、JavaScript 中的常量与变量等。

知识导图



2.1 数据类型

JavaScript 中的数据类型可以分为基本数据类型和复合数据类型。基本数据类型包括数值型（Number）、字符串型（String）、布尔型（Boolean）、空类型（Null）与未定义类型（Undefined）；复合数据类型包括对象、数组和函数等。关于复合数据类型将在后面的章节中学习，本节来学习 JavaScript 的基本数据类型。

2.1.1 数值型

数值型是 JavaScript 中最基本的数据类型。JavaScript 不是类型语言，与许多其他编程语言的不同之处在于，它不区分整型数值和浮点型数值。在 JavaScript 中，所有的数值都用浮点型来表示。

JavaScript 采用 IEEE 754 标准定义的 64 位浮点格式表示数字，它能表示的最大值为 $1.7976931348623157\times10^{308}$，最小值为 5×10^{-324}。在 JavaScript 中，数值有 3 种表示方式，分别为十进制、八进制和十六进制。

1. 十进制

默认情况下，JavaScript 中的数值为十进制显示。十进制整数是一个由 0 ～ 9 组成的数字序列。例如：

```
0
8
-9
100
```

2. 八进制

JavaScript 允许采用八进制格式来表示整型数据。八进制数据以数字 0 开头，其后是一个数字序列，这个序列中的数字必须在 0 和 7 之间，可以包括 0 和 7，例如：

```
07
0352
```

注意：虽然 JavaScript 支持八进制数据，但是有些浏览器却不支持，因此最好不要使用以 0 开头的整型数据。

3. 十六进制

在设置背景色或其他颜色时，颜色值一般采用十六进制数值表示。JavaScript 不但能够处理十进制数值，还能识别十六进制数值。在 JavaScript 代码中，常见的十六进制数值主要用来设置网页背景色、字体颜色等。

十六进制数值以“0X”或“0x”开头，其后紧跟十六进制数字序列。十六进制数值的数字序列可以是 0 ～ 9 中的某个数字，也可以是 a（A）～ f（F）中的某个字母。例如：

```
0xff
0XFF
0xCAFE
```

实例 1：输出十六进制颜色值对应的 RGB 值

一般情况下，网页中的颜色值以十六进制数字来表示。例如，颜色值 #CCFF99 中，十六进制数字 CC 表示 RGB 颜色中 R（红色）的值；FF 表示 RGB 颜色中 G（绿色）的值；99 表示 RGB 颜色中 B（蓝色）的值。下面编写程序，输出十六进制颜色值 #CCFF99 所对应的 RGB 值。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>十六进制颜色值对应的RGB值</title>
</head>
<body>
<script type="text/javascript">
    document.write("十六进制颜色值#CCFF99对应的RGB值：");//输出字符串
    document.write("<br>R："+0xCC);//输出R（红色）色值
    document.write("<br>G："+0xFF);//输出G（绿色）色值
    document.write("<br>B："+0x99);//输出B（蓝色）色值
    </script>
    </body>
    </html>
```

运行程序，结果如图 2-1 所示。

图 2-1　程序运行结果

4. 浮点型数据

浮点型数据的表示方法有两种：一种是传统记数法；另一种是科学记数法。下面分别进行介绍。

（1）传统记数法

传统记数法是将一个浮点数分为整数部分、小数点与小数部分。如果整数部分为 0，则可以将整数部分省略。例如：

```
3.1415
85.521
.231
```

（2）科学记数法

使用科学记数法表示浮点型数据的具体书写方式为：在实数后添加字母 e 或 E，然后再添加上一个带正号或负号的整数指数，其中正号可以省略。例如：

```
3e+5
3.14E11
1.231E-10
```

注意：在科学记数法中，e 或 E 后面的整数表示 10 的指数次幂，因此，这种表示方法所表示的数值为前面的实数乘以 10 的指数次幂。例如，3e+5 表示的数值为 3 乘以 10 的 5 次方，即 300000。

实例 2：输出以科学记数法表示的浮点数

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>输出科学记数法表示的浮点数</title>
</head>
<body>
<script type="text/javascript">
document.write ( "输出科学记数法表示的浮点数：" ) ;      //输出字符串
document.write ( "<p>" ) ;
//输出段落标记
document.write ( "科学记数法3e+5表示的浮点数为：" ) ;    //输出字符串
document.write ( 3e+5 ) ;
//输出浮点数
document.write ( "<br>" ) ;
//输出换行标记
document.write ( "科学记数法3.14e3表示的浮点数为：" ) ; //输出字符串
document.write ( 3.14e3 ) ;
//输出浮点数
document.write ( "<br>" ) ;
//输出换行标记
document.write ( "科学记数法1.212E-3表示的浮点数为：" ) ;    //输出字符串
document.write ( 1.212E-3 ) ;
   //输出浮点数
</script>
</body>
</html>
```

运行程序，结果如图2-2所示。

图 2-2　科学记数法表示的浮点数

5. 特殊值 Infinity

当数字运算结果超过 JavaScript 所能表示的数字上限（溢出）时，结果为一个特殊的无穷大（infinity）值，在 JavaScript 中以 Infinity 表示。同样，当负数的值超过 JavaScript 所能表示的负数范围时，结果为负无穷大，在 JavaScript 中以 -Infinity 表示。无穷大值的行为特性为：基于它们的加、减、乘和除运算结果还是无穷大。

实例 3：输出表达式 10/0 和 −10/0 的计算结果

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>输出10/0和-10/0的计算结果</title>
</head>
<body>
<script type="text/javascript">
document.write ( "输出10/0和-10/0的计算结果：" ) ;      //输出字符串
document.write ( "<p>" ) ;
//输出段落标记
document.write ( "10/0=" ) ;
//输出字符串
document.write ( 10/0 ) ;
//输出无穷大
document.write ( "<br>" ) ;
//输出换行标记
document.write ( "-10/0=" ) ;
//输出字符串
document.write ( -10/0 ) ;
//输出负无穷大
</script>
</body>
</html>
```

运行程序，结果如图 2-3 所示。

图 2-3　输出特殊值 Infinity

6. 特殊值 NaN

NaN 表示非数字值的特殊值，该属性用于指示某个值不是数字。在 JavaScript 中，当数学运算产生了未知的结果或错误时，就会返回 NaN，它表示该数学运算的结果是一个非数字值。例如，用 0 除以 0 的输出结果就是 NaN。

实例 4：输出 0/0 的计算结果

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>输出0/0的计算结果</title>
</head>
<body>
<script type="text/javascript">
document.write("0/0的计算结果");
//输出字符串
document.write("<p>");
//输出段落标记
document.write("0/0=");
//输出字符串
document.write(0/0);
//输出非数字值
</script>
</body>
</html>
```

运行程序，结果如图 2-4 所示。

图 2-4　输出 NaN

2.1.2　字符串型

字符串是由 0 个或者多个字符构成的，字符可以包括字母、数字、标点符号、空格或其他字符等，还可以包括汉字。在 JavaScript 中，字符串主要用来表示文本的数据类型。程序中的字符串型数据必须包含在单引号或双引号中。

（1）单引号括起来的字符串，代码如下：

```
'Hello JavaScript！'
'JavaScript@163.com'
'你好！JavaScript'
```

（2）双引号括起来的字符串，代码如下：

```
"Hello JavaScript！"
"JavaScript@163.com"
"你好！"
```

注意：空字符串不包含任何字符，也不包含任何空格，用一对引号表示，即""或''。另外，包含字符串的引号必须匹配，如果字符串前面用的是双引号，那么在字符串后面也必须用双引号，否则都使用单引号。

在编写 JavaScript 代码时，若字符串中有引号会产生匹配混乱的问题。例如，<img src="image5.jpg"> 这句代码，其中的双引号就有可能与字符串中的引号混淆，此时就需要使用转义字符。

JavaScript 中的转义字符是以一个反斜线（\）开始的。通过转义字符可以在字符串中添加不可显示的特殊字符，或者防止引号匹配混乱的问题。例如，字符串中的单引号可以使用“\'”来代替，双引号可以使用“\"”来代替。

```
"Hello\'JavaScript\'！"
```

JavaScript 中常用的转义字符如表 2-1 所示。

表 2-1 JavaScript 中的转义字符

序 号	转义字符	使用说明
1	\b	后退一格（Backspace），退格符
2	\f	换页（Form Feed）
3	\n	换行（New Line）
4	\r	回车（Carriage Return）
5	\t	水平制表符，Tab 空格
6	\'	单引号
7	\"	双引号
8	\\	反斜线（Backslash）
9	\v	垂直制表符
10	\xHH	十六进制整数，范围为 00 ～ FF
11	\uhhhh	十六进制编码的 Unicode 字符
12	\OOO	八进制整数，范围为 000 ～ 777

提示：转义字符“\n”在警告框中可以产生换行，但在 document.write() 语句中使用转义字符时，必须将其放在格式化文本块中才会起作用，所以脚本必须放置在 <pre> 和 </pre> 标签中。

实例 5：使用 alter() 语句输出一首古诗

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>输出一首古诗</title>
</head>
<body>
<script type="text/javascript">
alter("《春晓》\n\n春眠不觉晓，\n处处闻啼鸟。\n夜来风雨声，\n花落知多少。");
</script>
</body>
</html>
```

运行程序，结果如图 2-5 所示。

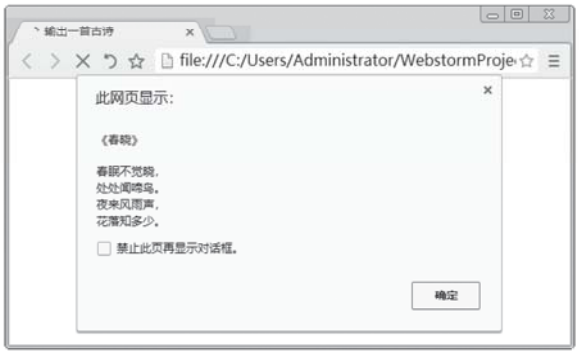


图 2-5 使用 alter() 语句输出

实例 6：使用 document.write() 语句输出一首古诗

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>输出一首古诗</title>
</head>
<body>
<script type="text/javascript">
document.write("<pre>");
document.write("《春晓》\n\n春眠不觉晓，\n处处闻啼鸟。\n夜来风雨声，\n花落知多少。");
document.write("</pre>");
</script>
</body>
</html>
```

运行程序，结果如图 2-6 所示。

图 2-6 使用 document.write() 语句输出

如果在代码中不添加 <pre></pre> 标签，则转义字符无效。代码如下：

```
document.write("《春晓》\n\n春眠不觉晓，\n处处闻啼鸟。\n夜来风雨声，\n花落知多少。");
```

在浏览器中的运行结果如图 2-7 所示。

图 2-7　转义字符无效

2.1.3　布尔型

在 JavaScript 中，布尔型数据类型只有两个值，一个是 true（真），一个是 false（假），它说明了某个事物是真还是假。通常，我们使用 1 表示真，0 表示假。布尔值通常在 JavaScript 程序中用来表示比较所得的结果。例如：

```
n==10
```

这句代码的作用是判断变量 n 的值是否和数值 10 相等，如果相等，比较的结果就是布尔值 true，否则结果就是 false。

布尔值通常用于 JavaScript 的控制结构。例如，JavaScript 的 if…else 语句，就是在布尔值为 true 时执行一个动作，而在布尔值为 false 时执行另一个动作。具体代码如下：

```
if(n==1)
   a=a+1;
else
   b=b+1;
```

这段代码的作用是首先判断 n 是否等于 1，如果相等，则执行 a=a+1，否则执行 b=b+1。

在 JavaScript 中，我们可以使用 Boolean（布尔对象）语句将非布尔值转换为布尔值（true 或者 false）。

实例 7：检查布尔对象是 true 还是 false

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>检查布尔对象是true还是false</title>
</head>
<body>
<script type="text/javascript">
var b1 = Boolean("");
//返回false，空字符串
var b2 = Boolean("s");
//返回true，非空字符串
var b3 = Boolean(0);
//返回false，数字0
var b4 = Boolean(1);
//返回true，非0数字
var b5 = Boolean(-1);
//返回true，非0数字
var b6 = Boolean(null);
//返回false
var b7 = Boolean(NaN); //返回false
document.write("空字符串是布尔值"+b1 + "<br>");
document.write("非空字符串是布尔值"+b2+ "<br>");
document.write("0为布尔值"+b3 +"<br>");
document.write("1为布尔值"+ b4 +"<br>");
document.write("-1为布尔值"+ b5 +"<br>");
```

```
    document.write ( "null是布尔值"+ b6+
"<br>" ) ;
    document.write ( "NaN是布尔值"+ b7
+"<br>" ) ;
    </script>
    </body>
    </html>
```

运行程序，结果如图 2-8 所示。

图 2-8　检查布尔型数据对象

2.1.4　未定义类型

undefined 是未定义类型的变量，表示变量还没有赋值，如 var a; 或者赋予一个不存在的属性值，如 var a=String.notProperty。

实例 8：使用未定义类型 undefined

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>输出未定义数据类型</title>
</head>
<body>
<script type="text/javascript">
var person;
document.write ( person + "<br />" ) ;
</script>
</body>
</html>
```

运行程序，结果如图 2-9 所示。

图 2-9　未定义数据类型 undefined

2.1.5　空类型

JavaScript 中的关键字 null 是一个特殊的值，表示空值，用于定义空的或不存在的引用。不过，null 不等同于空的字符串或 0。由此可见，null 与 undefined 的区别是：null 表示一个变量被赋予了一个空值，而 undefined 则表示该变量还未被赋值。

实例 9：使用 null 数据类型清空变量

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>使用null数据类型清空变量</title>
</head>
<body>
<script type="text/javascript">
var person;
var car="宝马";
document.write ( person + "<br>" ) ;
document.write ( car + "<br>" ) ;
var car=null
document.write ( car + "<br>" ) ;
</script>
</body>
</html>
```

运行程序，结果如图 2-10 所示。

图 2-10　使用空类型

2.2 JavaScript 的常量

在 JavaScript 中，常量与变量是数据结构的重要组成部分。其中，常量是指在程序运行过程中保持不变的数据。例如，123 是数值型常量，“Hello JavaScript！”是字符串常量，true 或 false 是布尔型常量。在 JavaScript 脚本编程中，这些数值是可以直接输入并使用的。

2.3 JavaScript 的变量

变量是相对于常量而言的。在 JavaScript 中，变量是指程序中一个命名的存储单元，它的主要作用就是为数据操作提供存放信息的容器。变量存储的数值是可以变化的，变量占据一段内存，通过变量的名字可以调用内存中的信息。

2.3.1 变量的命名

变量有两个基本特性，即变量名和变量值。为了便于理解，可以把变量看作一个贴有标签的抽屉，标签的名字就是这个变量的名字，而抽屉里面的东西就相当于变量的值。对于变量的使用，必须明确变量的名称、变量的声明、变量的赋值以及变量的类型。

在 JavaScript 中，变量的命名规则如下。

（1）必须以字母或下划线开头，其他字符可以是数字、字母或下划线。例如，txtName 与 _txtName 都是合法的变量名，而 1txtName 和 &txtName 都是非法的变量名。

（2）变量名只能由字母、数字、下划线组成，不能包含空格、加号、减号等符号，不能用汉字做变量名。例如，txt%Name、名称文本、txt-Name 都是非法变量名。

（3）JavaScript 的变量名是区分大小写的。例如，Name 与 name 代表两个不同的变量。

（4）不能使用 JavaScript 中的保留关键字作为变量名。例如，var、enum、const 都是非法变量名。JavaScript 中的保留关键字如表 2-2 所示。

表 2-2 JavaScript 中的保留关键字

abstract	arguments	boolean	break	byte	case
catch	char	class	const	continue	debugger
default	delete	do	double	else	enum
eval	export	extends	false	final	finally
float	for	function	goto	if	implements
import	in	instanceof	int	interface	let
long	native	new	null	package	private
protected	public	return	short	static	super
switch	synchronized	this	throw	throws	transient
true	try	typeof	var	void	volatile
while	with	yield			

提示：JavaScript 中的保留关键字是指在 JavaScript 语言中有特定含义，并成为 JavaScript 语法中一部分的那些字。JavaScript 中的保留关键字不可以用作变量、标签或者函数名。

在给变量命名时，最好使用便于记忆、有意义的名称，以增加程序的可读性。例如，定义一个用于描述人物的变量，可以命名为 person。

2.3.2 变量的声明

尽管 JavaScript 是一种弱类型的脚本语言，变量可以在不声明的情况下直接使用，但在实际使用过程中，最好还是先使用 var 关键字对变量进行声明。语法格式如下：

```
var variablename;
```

variablename 为变量名。例如，声明一个变量 car，代码如下：

```
var car;
```

可以使用一个关键字 var 同时声明多个变量，例如：

```
var x,y;
```

此语句就同时声明了x和y两个变量。

注意：声明 JavaScript 变量时，不指定变量的数据类型。一个变量一旦声明，可以存放任何数据类型的信息，JavaScript 会根据存放的信息类型，自动为变量分配合适的数据类型。

2.3.3 变量的赋值

在声明变量的同时可以为变量赋值，这一过程也被称为变量初始化，例如：

```
var username="杜牧";   //声明变量并进行初始化赋值
var x=5,y=12;          //声明多个变量并进行初始化赋值
```

这里声明了 3 个变量 username、x 和 y，并分别对其进行了赋值。
另外，还可以在声明变量之后再对变量进行赋值，例如：

```
var username;          //声明变量
username="杜牧";       //对变量进行赋值
```

在 JavaScript 中，可以不先声明变量而直接对其进行赋值。例如，给一个未声明的变量赋值，然后输出这个变量的值，代码如下：

```
username="杜牧";                 //未声明变量就对变量进行赋值
document.write(username);        //输出变量的值
```

程序的运行结果如下：

```
杜牧
```

注意：虽然 JavaScript 允许给一个未声明的变量直接进行赋值，但还是建议在使用变量之前先对其进行声明，这是因为 JavaScript 采用动态编译方式，而这种方式不易发现代码中的错误，特别是变量命名方面的错误。

在使用变量时，最容易忽略的就是字母大小写。例如下面的代码。

```
var name="杜牧";
document.write(Name);
```

在运行这段代码时，就会出现错误，这是因为定义了一个变量name，而在输出语句中书写的是“Name”，这就忽略了字母的大小写。

在 JavaScript 中，如果只是声明了变量，并未对其赋值，其值默认为 undefined。如果出现重复声明的变量，且该变量已有一个初始值，那么后来声明的变量相当于给变量重新赋值。

实例 10：变量赋值的应用

声明一个未赋值的变量 a，重复声明一个变量 b，并输出这两个变量的值，代码如下：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>变量赋值的应用</title>
</head>
<body>
<script type="text/javascript">
var a;    //声明变量a未赋值
var b="Hello JavaScript!";
//声明变量b并对其进行赋值
var b="你好，JavaScript!";
//重复声明变量b并对其进行赋值
document.write(a);
//输出变量a的值
document.write("<br>");
//输出换行标签
document.write(b);
//输出变量b的值
</script>
</body>
</html>
```

运行程序，结果如图 2-11 所示。

图 2-11　变量赋值的应用

2.3.4　变量的类型

变量的类型是指变量的值所属的数据类型，它可以是数值型、字符串型、布尔型等，因为 JavaScript 是一个弱类型的程序语言，所以可以把任意类型的数据赋值给变量。

例如，先将一个数值型数据赋值给一个变量，在程序运行过程中，还可以将一个字符串型数据赋值给同一个变量。代码如下：

```
var b=3.14;                    //声明变量b并对其进行赋值
b="你好，JavaScript!";          //将字符串型数据赋值变量b
document.write(b);             //输出变量b的值
```

上述代码的运行结果如图 2-12 所示。

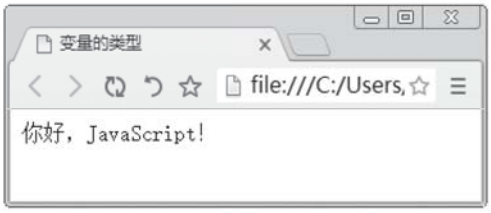


图 2-12　程序运行结果

2.3.5 变量的作用域

变量的作用范围又称为作用域，是指某变量在程序中的有效范围。根据作用域的不同，变量可划分为全局变量和局部变量。

（1）全局变量：全局变量的作用域是全局性的，即在整个 JavaScript 程序中，全局变量处处都存在。

（2）局部变量：局部变量是函数内部声明的，只作用于函数内部，其作用域是局部性的；函数的参数也是局部性的，只在函数内部起作用。

在函数内部，局部变量的优先级高于同名的全局变量。也就是说，如果存在与全局变量名称相同的局部变量，或者在函数内部声明了与全局变量同名的参数，则该全局变量将不再起作用。

实例 11：变量作用域的应用

定义一个全部变量与一个局部变量，然后输出变量的作用域类型。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>变量的作用域</title>
</head>
<body>
<script type="text/javascript">
var scope="全局变量"; //声明一个全局变量
function checkscope()
{
var scope="局部变量";
//声明一个同名的局部变量
document.write(scope);
//使用的是局部变量，而不是全局变量
}
checkscope();
//调用函数，输出结果
</script>
</body>
</html>
```

运行程序，结果如图 2-13 所示。从结果中可以看出，输出的是“局部变量”，这就说明局部变量的优先级高于同名的全局变量。

图 2-13 变量的优先级

注意：虽然在全局作用域中可以不使用 var 声明变量，但在声明局部变量时，一定要使用 var。

2.4 新手常见疑难问题

疑问 1：JavaScript 中有哪些基本数据类型?

JavaScript 中有 5 个简单数据类型，也被称为基本数据类型，分别是 Undefined、Null、Boolean、Number 和 String。还包含多个复杂数据类型，分别是 Object、Array、Function 等。

疑问 2：在 JavaScript 中，可以直接使用变量吗?

不能。在 JavaScript 中的变量必须先定义才能使用，没有定义过的变量是不能直接使用的。例如，直接输出一个未定义的变量，代码如下：

```
document.write(x);                    //输出未定义的变量x的值
```

这里由于没有定义变量 x，却使用 document.write 语句直接输出它的值，这就会发生错误。

2.5 实战技能训练营

实战 1：使用变量输出个人基本信息

定义用于存储个人信息的变量，然后输出这些个人信息，运行结果如图 2-14 所示。

实战 2：输出《水调歌头·明月几时有》

使用 document.write() 语句和转义字符输出诗词内容，运行结果如图 2-15 所示。

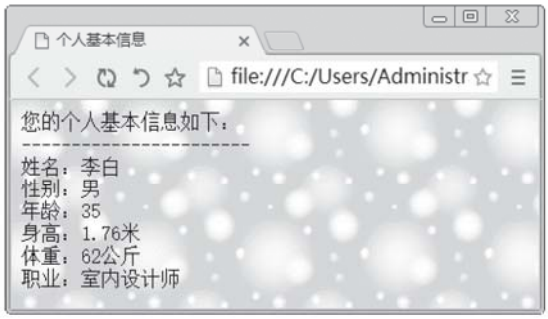


图 2-14　输出个人信息

图 2-15　输出诗词内容