函数、方法和对象

管理科学与工程学科
耿方方

主要内容

- 函数和方法
- 对象
- 内置对象

浏览器需要非常详细的指令才能知道怎么做。程序员使用函数、方法和对象来组织代码。

函数和方法

函数是什么？

- 函数和方法：函数是由一系列语句组成，这些语句因为执行特定的任务而被分到一组。方法的功能与函数一样，只是方法在对象内部创建。
- 如果需要多次使用同一段代码，可以把它们封装成一个函数，函数就是一组允许在你的代码里随时调用的语句，每个函数实际上就是一个短小的脚本。
- 一个简单的函数如下所示：

```
function show(){
    var myArray=["BMW","Volvo","Saab","Ford"];
    for (var i=0;i<myArray.length;i++){
        alert(myArray[i]);
    }
}
```

- 这个函数将循环输出数组中的内容。现在如果想在自己的脚本中执行这一动作，可以随时调用如下语句来执行这个函数：

```
show();
```

函数和方法

函数是什么？

- 案例1:

```
<html>
    <head>
        <title>一个简单的函数</title>
        <meta charset="UTF-8">
    </head>
    <body>
        <h1>一个简单的函数</h1>
    <div id="message">Welcome to our site!</div>
    <script src="demo4-01-1.js"></script>
    </body>
</html>
```

```
var msg = 'Sign up to receive our
newsletter for 10% off!';

function updateMessage() {
  var el =
document.getElementById('message');
  el.textContent = msg;
}
updateMessage();
```

函数和方法

如何声明函数？

- 在JavaScript中，函数的定义是由关键字funciton、函数名、函数的参数及置于大括号中的函数体组成的。定义函数的基本语法如下：

- funciton functionName([parameter1,parameter2,…]){

- statements;

- [return expression;]}

- functionName:必选，用于指定函数名。在同一个页面中，函数名必须是唯一的。
- parameter: 可选，用于指定参数列表。一个函数最多有255个参数。
- statements:必选，是函数体，用于实现函数功能的语句。
- expression: 可选，用于返回函数值。

"function" 这个词必须是小写的，否则 JavaScript 就会出错。另外需要注意的是，必须使用大小写完全相同的函数名来调用函数。

函数和方法

如何声明函数？

- 有时，函数需要特定的信息来执行任务。在这种情况下，声明函数时要给它提供形参。在函数内部形参类似于变量。
- ```
  function getArea(width,height){
      return width*height;}
  ```
- 调用带有形参的函数时，需在函数名后面的括号中指定一些值。这些值就是实参，可以象变量一样赋值。
- 值作为实参
- getArea (3, 4)
- 变量作为实参
- ```
  wallWidth=3;
  wallHeight=4;getArea(wallWidth,wallHeight);
  ```

函数和方法

如何声明函数？

- return语句
- return 语句用来规定从函数返回的值。因此，需要返回某个值的函数必须使用这个 return 语句。
- 调用函数的返回值只需将函数赋值给一个变量，然后调用该变量即可；
- 函数在执行过程中只要执行完return语句就会停止继续执行函数体中的代码，因此return语句后的代码都不会执行。

函数和方法

如何声明函数？

- 实例2
- <html>
- <head>
- <title>TODO supply a title</title>
- <meta charset="UTF-8">
- <script type="text/javascript">
- function product(a,b) {
- return a*b }
- </script>
- </head>
- <body>
- <script type="text/javascript">
- document.write(product(6,5))
- </script>
- body 部分中的脚本调用一个带有两个参数 (6 和 5) 的函数。
- 该函数会返回这两个参数的乘积。
- </body>
- </html>

函数和方法

如何声明函数？

- 内存和变量的工作原理
- 全局变量使用更多地内存。浏览器需要在页面载入期间保存它们。局部变量只需在函数执行期间被保存。
- 在代码中创建变量：每个变量的声明都存在内存开销。浏览器需要保存的变量越多，运行脚本时需要用到的内存资源越多。所需资源越多脚本运行越慢，即页面响应用户的时间越长。
- 例：`var width=15;`
- `var height=30;`
- `var isWall=true;`
- `var  canPaint=true;`

相同的值使用同一内存存储，可用于不同的变量。

函数和方法

如何声明函数？

- 命名冲突：
- 很多人共同编写脚本时，尽量使用局部变量。如果脚本中存在同名的全局变量，将可能导致错误。
- 全局作用域中的变量：有命名冲突。
- 函数作用域中的变量：彼此之间没有冲突。
- 例：
- Function showPlotSize(){

  Var width=3; height=2; return ‘area:’ +(width*height);

} var msg=showPlotSize();

- Function showGardenSize{

  Var width=12; height=30; return width*height;

- } var msg=showGardenSize();

函数和方法

如何声明函数？

- 立即调用函数表达式：使用右花括号的最后一对括号告诉解释器马上调用此函数。分组操作符的括号确保解释器将这作为一个表达式对待。
- 案例3：

```
<html>
    <head>
        <title>立即调用函数表达式</title>
        <meta charset="UTF-8">
</head>
    <body>
        <div>立即调用函数表达式</div>
        <script>
             var area=(function(){var width=3;
             var height=5;
            return width*height;}());
            document.write(area);
         </script>
    </body>
</html>
```

函数和方法

函数的调用？

- 1、简单的函数调用

```
<html>
    <head>
        <script>
            function functionName(parameter){
                statements;}
        </script>
    </head>
    <body>
        <script>
                functionName(parameter);
        </script>
    </body>
</html>
```

函数和方法

- 2、通过链接调用函数
- 函数除了可以在响应事件中被调用外，还可以在链接中被调用，在<a>标记中的href属性中使用“javascript关键字”格式调用函数，用户单击这个超链接时，相关函数将被执行。

```
<script>
   function test(){
       alert(“I like JavaScript”);}
</script>
<body>
    <a href=“javascript:test();”>通过链接调用函数</a>
</body>
```

函数和方法

函数的调用？

- 3、在响应事件中调用函数
- 用户单击某个按钮或选中某个复选框时都将触发事件，通过编写程序对事件做出反应的行为称为响应事件，在JavaScript语言中，将函数与事件联系在一起就完成了响应事件的过程。

```
<script>
    function test(){
        alert(“I like JavaScript”);}
</script>
<body>
    <form action=“” method=“post” name=“form1”>
     <input type=“button” value=“提交” onClick=“test();”>
    </form>
</body>
```

函数和方法

函数的类型？

- 1、函数直接量
- 函数直接量是一个表达式，它可以定义一个匿名函数。函数直接量的语法和function语句非常相似，只不过它被用作表达式，而不是用作语句，而且也无需指定函数名。
- 下面两行代码分别使用function语句和直接量定义了两个基本上相同的函数：
- `function f(x){return x*x;}`
- `var f=function(x){return x*x;}`
- 函数直接量都是由JavaScript的表达式创建的，而不是由语句创建的，所以使用它们的方式也就更加灵活，尤其适用于那些只使用一次而且无需命名的函数。

函数和方法

函数的类型？

- 2、构造函数
- 使用new Object ()来创建一个新的空对象。还可以使用
- var array=new Array (10) ;
- var today=new Date();来创建其他类型的JS对象。new运算符后面必须跟着一个函数调用，new创建了一个新的没有任何属性地对象，然后调用该函数，把新的对象作为this关键字的值传递。设计来和new运算符一起使用的函数叫做构造函数。
- 构造函数的工作是初始化一个新创建的对象，设置在使用对象前需要设置的所有属性。可以编写自己的构造函数，只需要编写一个为this添加属性的函数就可以了。例如:

函数和方法

函数的类型？

- 2、构造函数
- function Rectangle(w,h){
- 　　this.width=w;
- 　　this.height=h;
- }
- var rect1=new Rectangle(2,4);
- var rect2=new Rectangle(8.5,11);
- 通常定义一个适当的构造函数，就定义了对象的一个类；构造函数通常没有返回值。它们初始化为this的值来传递的对象，并且没有返回值。然而，一个构造函数是允许返回一个对象值的。返回的对象成为new表达式的值。

函数和方法

函数的类型？

- 3、嵌套函数
- 所谓嵌套函数，即在函数内部再定义一个函数，这样定义的优点在于可以使内部函数轻松获得外部函数的参数及函数的全局变量等。

```
<script>
    var outter=10;
    function functionName(parameters1,parameters2){
        function InnerFunction(){
         somestatements;}}
</script>
```

函数和方法

函数的类型？

- 4、递归函数
- 所谓递归函数，就是函数在自身的函数体内调用自身。使用递归函数时一定要注意，处理不当会使程序进入死循环，它只在特定情况下使用，例如处理阶乘问题时。
- <script>
- var outter=10;
- function functionName(parameters1){
- functionName(parameters2); }
- </script>

函数和方法

函数的类型？

- 4、递归函数
- 例如：<html>
- <head>
- <title>递归函数的应用</title>
- <script>
- function f(num){
- if(num<=1){
- return 1;}
- else{
- return f(num-1)*num;}}
- </script>
- </head>
- <body>
- <script>
- alert(“10!的结果为：”+f (10));
- <script>
- </body>
- </html>

函数和方法

函数的类型？

- 例如：函数的实例

```
<script>
    function checkCode(digit){
            var result='';
            for(i=0;i<parseInt(digit);i++;){
             result=result+(parseInt(Math.random()*10)).toString();}
  return result;}
    function deal(){
          result.innerHTML=”  产  生  的  验  证  码  :  ”+checkCode
 (form1.digit.value) ;}


</script>
```

对象

- 对象是一种非常重要的数据类型，对象是自包含的数据集合，包含在对象里的数据可以通过两种形式访问，属性和方法：属性是隶属于某个特定对象的变量；方法是只有某个特定对象才能调用的函数。
- 对象就是一些由一些属性和方法组合在一起而构成的一个数据实体。

对象

- 例:
- var hotel={
- name: ‘Quay’ ,
- rooms:40,
- booked:25,
- roomTypes:[‘twin’ ,’ double’ ,’ suite’],
- checkAvailability:function(){
- return this.rooms=this.booked;
- }
- };

右边的例子展示了一个酒店，它包含了四个属性和一个方法。在对象中，名称被称为键。属性的值可以是字符串、数字、布尔值、数组或是对象。方法的值只能是函数。

Hotel对象包含的键/值对：

属性：键	值
name	字符串
rooms	数字
booked	数字
roomTypes	数组
方法 checkAvailability	函数

对象

- 创建对象的方法
- 1、字面量语法：对象是花括号以及其中的内容，存储于变量hotel中，称为hotel对象。每个键和值之间用冒号分隔。每个属性和方法之间用冒号分割。

```
var hotel={
    name:  ‘Quay’ ,
    rooms:40,
    booked:25,
    roomTypes:[ ‘twin’ ,’ double’ ,’ suite’ ],
checkAvailability:function(){
  return this.rooms=this.booked;
}
};
```

对象

- 访问对象的方法

- 在JavaScript里，属性和方法都使用“点”语法来访问，其用法如下所示：

```
object.property
object.method()
```

- 例：
- var hotelName=hotel.name;
- var roomsFree=hotel.checkAvailability();

对象

- 案例4：

```
<html>
    <head>
        <title>hotel availability</title>
        <meta charset="UTF-8">
    </head>
    <body>
        <h2>hotel availability</h2>
      <div id="hotelName"></div>
      <div id="availability">
        <p id="rooms"></p>
        <p>rooms left</p>
      </div>
    <script src="demo4-04-1.js"></script>
    </body>
</html>
```

```
var hotel = {
  name : 'Quay',
  rooms : 40,
  booked : 25,
  checkAvailability : function() {
    return this.rooms - this.booked; }
};
var elName =
document.getElementById('hotelName');
elName.textContent = hotel.name;
var elRooms =
document.getElementById('rooms');
elRooms.textContent =
hotel.checkAvailability();   // Update HTML
with property of the object
```

对象

- 2、构造函数方法：
- New关键字和对象的构造函数相结合可创建一个空白对象，随后可以为其添加属性和方法。
- 使用new关键字和Object ()构造函数联合创建一个新对象。创建空白对象以后，可以使用点语法添加属性和方法。每个创建属性和方法的行都应该以分号结束。
- 例：

```
var hotel=new object();
 hotel.name:  ‘Quay’ ,
 hotel.rooms:40,
 hotel.booked:25,
 hotel.roomTypes:[ ‘twin’ ,’ double’ ,’ suite’ ],
hotel.checkAvailability:function(){
  return this.rooms=this.booked;
}
```

对象

- 2、构造函数方法：创建很多对象。

对象构造函数可以使用函数作为模板来创建对象。首先，创建带有对象属性和方法的模板。

例：

```
function Hotel(name,rooms,booked){
this.name=name;
this.rooms=rooms;
this.booked=booked;
hotel.checkAvailability:function(){
return this.rooms=this.booked;}
}
```

一个名称为Hotel的函数作为模板，用来创建表示酒店的对象。构造函数的名称通常为大写。

对象

- 2、构造函数方法：创建很多对象。

使用构造函数创建对象的实例，new关键字后紧接着调用创建新对象的函数，每个对象的属性作为实参传递给函数。

```
var quayHotel=new Hotel( ‘Quay’ ,40,25);

var parkHotel=new Hotel( ‘Park’ ,120,77);
```

对象

- 案例5:

```
<html>
 <head>
    <title>创建对象-构造函数</title>
    <meta charset="UTF-8">
</head>
 <body>
    <h1>TravelWorthy</h1>
 <div id="info">
   <h2>hotel availability</h2>
   <div id="hotelName"></div>
   <div id="availability">
     <p id="rooms"></p>
     <p>rooms left</p>
   </div>
 </div>
 <script src="demo4-05-1.js"></script>
 </body>
</html>
```

```
var hotel = new Object();

hotel.name = 'Park';
hotel.rooms = 120;
hotel.booked = 77;
hotel.checkAvailability = function() {
  return this.rooms - this.booked;
};

var elName =
document.getElementById('hotelName'); //
Get element
elName.textContent = hotel.name; var
elRooms =
document.getElementById('rooms');    // Get
element
elRooms.textContent =
hotel.checkAvailability();
```

对象

- 案例6：

```
<html>
 <head>
     <title>创建对象-构造函数2</title>
     <meta charset="UTF-8">
</head>
 <body>
     <h1>TravelWorthy</h1>
 <div id="info">
   <h2>hotel availability</h2>
   <div id="hotel1"></div>
   <div id="hotel2"></div>
 </div>
 <script src="demo4-06-1.js"></script>
 </body>
</html>
```

```
function Hotel(name, rooms, booked) {
  this.name = name;
  this.rooms = rooms;
  this.booked = booked;
  this.checkAvailability = function() {
    return this.rooms - this.booked;
  };
}
var quayHotel = new Hotel('Quay', 40, 25);
var parkHotel = new Hotel('Park', 120, 77);
var details1 = quayHotel.name + ' rooms: ';
    details1 += quayHotel.checkAvailability();
var elHotel1 =
document.getElementById('hotel1');
elHotel1.textContent = details1;
var details2 = parkHotel.name + ' rooms: ';
    details2 += parkHotel.checkAvailability();
var elHotel2 =
document.getElementById('hotel2');
elHotel2.textContent = details2;
```

对象

- 添加和删除属性
- 例如:
- Var hotel={

  name:’ park’ ;

  rooms:120,

  booked:77
- };
- 可以通过

  hotel.pool=false;

  添加属性;

可以通过

  delete hotel.booked;

删除属性;

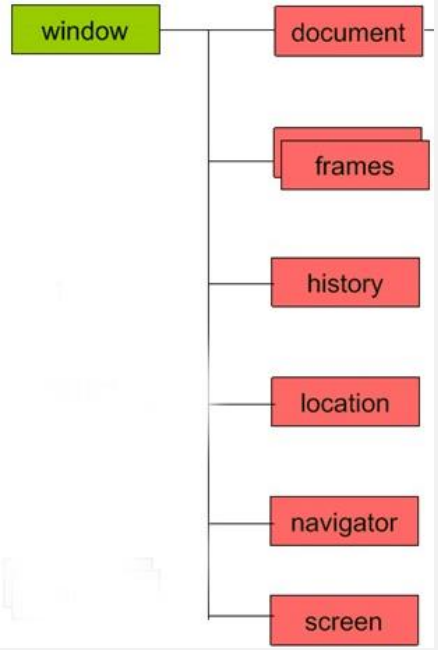
对象

- 浏览器附带了一系列内置对象，带包当前窗口中网页的一些内容。这些内置对象的行为类似于用于创建交互式网页的工具。
- 共有三组内置对象：浏览器对象模型、文档对象模型、全局JavaScript对象模型

对象

浏览器对象模型

- 浏览器对象代表当前浏览器窗口或标签，位于浏览器对象模型中的最顶端，其中包含了描述浏览器的对象。



3. location对象

location对象提供了与当前窗口中**加载的文档有关的信息以及一些导航功能;**

它既是window对象的属性，同时也是document对象的属性.

主要属性有：

属性	例如	说明
href	http://www.nba.com	返回当前完整的URL地址==location.toString()
host	www.nba.com:80	返回服务器名称和端口号
hostname	www.nba.com	返回服务器名称
port	8080	返回URL中指定的端口号,如果没有,返回空字符串
pathname	/games/2015	返回URL中的目录和文件名
protocol	http:	返回页面使用的协议，通常是http:或https:

对象

- 浏览器对象代表当前浏览器窗口或标签，位于浏览器对象模型中的最顶端，其中包含了描述浏览器的对象。

方法	例如	说明
search	？username=Games	返回URL的查询字符串，这个字符串以问号开头
assign()	location.assign(url)	立即打开新URL并在浏览器历史中生成一条记录，相当于直接设置location.href值， 也可以修改location对象的其它属性来重新加载
replace()	location.replace(url)	打开新URL，但是不会生成历史记录，使用replace()之后，用户不能通过“后退”回到前一个页面
reload()	location.reload([true])	重新加载当前页面，不传递参数时会以最有效方式加载（可能从缓存中加载）， 传入true时，则强制从浏览器重新加载

对象

- 案例7：

```
<html>
    <head>
        <title>当前网页信息</title>
        <meta charset="UTF-8">
</head>
    <body>
        <h1>TravelWorthy</h1>
    <div id="info"></div>
    <script src="demo4-07-1.js"></script>
    </body>
</html>
```

```
var msg = '<h2>browser
window</h2><p>width: ' +
window.innerWidth + '</p>';
msg += '<p>height: ' + window.innerHeight
+ '</p>';
msg += '<h2>history</h2><p>items: ' +
window.history.length + '</p>';
msg += '<h2>screen</h2><p>width: ' +
window.screen.width + '</p>';
msg += '<p>height: ' +
window.screen.height + '</p>';

var el = document.getElementById('info');
el.innerHTML = msg;
alert('Current page: ' + window.location)
```

对象

- 文档对象模型的最顶端是document对象。它代表当前浏览器窗口或标签中载入的页面。

属性	描述
document.title	当前文档的标题
document.lastModified	文档最后一次被修改的日期
document.URL	返回包含当前文档URL的字符串
document.domain	返回当前文档的域

对象

- 文档对象模型的最顶端是document对象。它代表当前浏览器窗口或标签中载入的页面。

方法	描述
document.write()	将文本写入文档
document.getElmentById()	返回与id属性值相匹配的元素
document.querySelectorAll()	返回一组元素
document.createElement（）	创建新元素
Document.createTextNode()	创建新的文本节点

对象

- 案例8：

```
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
</head>
    <body>
       <h1>TravelWorthy</h1>
    <div id="footer"></div>
    <script src="demo4-08-1.js"></script>
    </body>
</html>
```

```
var msg = '<p><b>page title: </b>' +
document.title + '<br />';
msg += '<b>page address: </b>' +
document.URL + '<br />';
msg += '<b>last modified: </b>' +
document.lastModified + '</p>';

var el = document.getElementById('footer');
el.innerHTML = msg;
```

对象

- 基本全局对象

  String　　用于处理字符串

  Number　　用于处理数字

  Boolean　　用于处理布尔值

- 处理真实世界的概念:

  Date　　展示和处理日期

  Math　　用于处理数字和计算

  RegExp　　用于处理匹配文字的字符串模式

对象

全局对象：String

- String对象
- 示例：var saying=‘Home sweet home ’ ;字符串分配的索引编号是从0开始计数的。

属性	描述	示例	结果
length	返回字符串中的字符数	saying.length	16

方法	描述	示例	结果
toUpperCase （）	将字符串修改为大写字母	saying. toUpperCase （）	‘HOME SWEET HOME ’
toLowerCase （）	将字符串修改为小写字母	saying. toLowerCase()	‘home sweet home’
charAt （）	以索引为编号，返回字符	saying. charAt(12)	‘o’
indexOf()	返回出现字符的索引编号	saying. indexOf(‘ee’)	7
lastIndexOf()	返回最后出现该字符的索引编号	Saying.lastIndexOf(‘e’)	14
substring()	返回包括两个索引编号之间的字符，包括前面索引的字符，不包括后面索引的字符。	Saying.substring(8,14)	‘et hom’
split()	将字符串分割，存储在一个数组中	Saying.split(‘ ’)	[‘Home’,’sweet’,’home’]
trim()	删除字符串开始和结尾的空格	Saying.trim()	‘Home sweet home’
replace()	查找替换	Saying.replace(‘me’,’w’)	‘How sweet home’

对象

案例9：

```
<html>
    <head>
        <title>字符串对象</title>
        <meta charset="UTF-8">
</head>
    <body>
        <h1>TravelWorthy</h1>
    <div id="info"></div>
    <script src="demo4-09-1.js"></script>
    </body>
</html>
```

全局对象：String

```
var saying = 'Home sweet home ';
var msg = '<h2>length</h2><p>' +
saying.length + '</p>';
msg += '<h2>uppercase</h2><p>' +
saying.toUpperCase() + '</p>';
msg += '<h2>lowercase</h2><p>' +
saying.toLowerCase() + '</p>';
msg += '<h2>character index: 12</h2><p>' +
saying.charAt(12) + '</p>';
msg += '<h2>first ee</h2><p>' +
saying.indexOf('ee') + '</p>';
msg += '<h2>last e</h2><p>' +
saying.lastIndexOf('e') + '</p>';
msg += '<h2>character index: 8-14</h2><p>'
+ saying.substring(8, 14) + '</p>';
msg += '<h2>replace</h2><p>' +
saying.replace('me', 'w') + '</p>';
element whose id attribute has a value of info
var el = document.getElementById('info');
el.innerHTML = msg;
```

对象

- 数字对象

方法	描述
isNaN()	检查是否为数字
toFixed()	将特定数字四舍五入至指定小数位数（返回一个字符串）
toPrecision()	按数字位数四舍五入（返回一个字符串）
toExponential()	以字符串的形式返回指数计数法表示的数字

对象

全局对象：数字对象

案例10：

```
<html>
    <head>
        <title>数字对象</title>
        <meta charset="UTF-8">
</head>
    <body>
        <h1>TravelWorthy</h1>
    <div id="info"></div>
    <script src="demo4-10-1.js"></script>
    </body>
</html>
```

```
var originalNumber = 10.23456;
var msg = '<h2>original number</h2><p>' +
originalNumber + '</p>';
msg += '<h2>3 decimal places</h2><p>' +
originalNumber.toFixed(3) + '</p>';
msg += '<h2>3 digits</h2><p>' +
originalNumber.toPrecision(3) + '</p>';

var el = document.getElementById('info');
el.innerHTML = msg;
```

对象

全局对象：Math

- Math对象

属性	描述
Math.PI	返回pi（3.1415926）

方法	描述
Math.round	把数四舍五入为最接近的整数
Math.sqrt(n)	返回数的平方根。
Math.ceil()	对数进行上舍入至离它最近并大于它本身的整数
Math.floor()	对数舍入至离它最近并小于它本身的整数
Math.random()	获取一个从0（包括）到1（不包括的）随机数

对象

全局对象：Math

案例11:

```
<html>
    <head>
        <title>获取随机数</title>
        <meta charset="UTF-8">
    </head>
    <body>
        <h1>TravelWorthy</h1>
    <div id="info"></div>
    <script src="demo4-11-1.js"></script>
    </body>
</html>
```

```
var randomNum = Math.floor((Math.random()
* 10) + 1);
var el = document.getElementById('info');
el.innerHTML = '<h2>random
number</h2><p>' + randomNum + '</p>';
```

对象

全局对象：Date

- Date对象

为了创建日期，需首先创建Date对象的一个实例，然后对其指定需要展示的时间和日期。

```
var today=new Date();
```

方法	描述
Date()	返回当日的日期和时间。
getDate()	从 Date 对象返回一个月中的某一天 (1 ~ 31)。
getDay()	从 Date 对象返回一周中的某一天 (0 ~ 6)。
getMonth()	从 Date 对象返回月份 (0 ~ 11)。
getFullYear()	从 Date 对象以四位数字返回年份。
getYear()	请使用 getFullYear() 方法代替。
getHours()	返回 Date 对象的小时 (0 ~ 23)。
getMinutes()	返回 Date 对象的分钟 (0 ~ 59)。
getSeconds()	返回 Date 对象的秒数 (0 ~ 59)。
getMilliseconds()	返回 Date 对象的毫秒(0 ~ 999)。
getTime()	返回 1970 年 1 月 1 日至今的毫秒数。
getTimezoneOffset()	返回本地时间与格林威治标准时间 (GMT) 的分钟差。
getUTCDate()	根据世界时从 Date 对象返回月中的一天 (1 ~ 31)。
getUTCDay()	根据世界时从 Date 对象返回周中的一天 (0 ~ 6)。
getUTCMonth()	根据世界时从 Date 对象返回月份 (0 ~ 11)。
getUTCFullYear()	根据世界时从 Date 对象返回四位数的年份。
getUTCHours()	根据世界时返回 Date 对象的小时 (0 ~ 23)。
getUTCMinutes()	根据世界时返回 Date 对象的分钟 (0 ~ 59)。
getUTCSeconds()	根据世界时返回 Date 对象的秒钟 (0 ~ 59)。
getUTCMilliseconds()	根据世界时返回 Date 对象的毫秒(0 ~ 999)。

对象

- Date对象

setDate()	设置 Date 对象中月的某一天 (1 ~ 31)。
setMonth()	设置 Date 对象中月份 (0 ~ 11)。
setFullYear()	设置 Date 对象中的年份（四位数字）。
setYear()	请使用 setFullYear() 方法代替。
setHours()	设置 Date 对象中的小时 (0 ~ 23)。
setMinutes()	设置 Date 对象中的分钟 (0 ~ 59)。
setSeconds()	设置 Date 对象中的秒钟 (0 ~ 59)。
setMilliseconds()	设置 Date 对象中的毫秒 (0 ~ 999)。
setTime()	以毫秒设置 Date 对象。
setUTCDate()	根据世界时设置 Date 对象中月份的一天 (1 ~ 31)。
setUTCMonth()	根据世界时设置 Date 对象中的月份 (0 ~ 11)。
setUTCFullYear()	根据世界时设置 Date 对象中的年份（四位数字）。
setUTCHours()	根据世界时设置 Date 对象中的小时 (0 ~ 23)。
setUTCMinutes()	根据世界时设置 Date 对象中的分钟 (0 ~ 59)。
setUTCSeconds()	根据世界时设置 Date 对象中的秒钟 (0 ~ 59)。
setUTCMilliseconds()	根据世界时设置 Date 对象中的毫秒 (0 ~ 999)。
toSource()	返回该对象的源代码。
toString()	把 Date 对象转换为字符串。
toTimeString()	把 Date 对象的时间部分转换为字符串。
toDateString()	把 Date 对象的日期部分转换为字符串。
toGMTString()	请使用 toUTCString() 方法代替。
toUTCString()	根据世界时，把 Date 对象转换为字符串。
toLocaleString()	根据本地时间格式，把 Date 对象转换为字符串。
toLocaleTimeString()	根据本地时间格式，把 Date 对象的时间部分转换为字符串。

对象

全局对象：Date

案例12:

```
<script language="javascript">
        var now=new Date();
        var year=now.getFullYear();
        var month=now.getMonth();
        var date=now.getDate();
        var day=now.getDay();
        var hour=now.getHours();
        var minu=now.getMinutes();
        var sec=now.getSeconds();
        var week;
        month=month+1;
        if(month<10) month="0"+month;
        if(date<10) date="0"+date;
        if(hour<10) hour="0"+hour;
        if(minu<10) minu="0"+minu;
        if(sec<10) sec="0"+sec;
        var arr_week=new Array("星期日","星期一","星期二","星期三","星期四","星期五","星期六");
        week=arr_week[day];
        var time="";
        time=year+"年"+month+"月"+date+"日 "+week+" "+hour+":"+minu+":"+sec;
    document.write("当前时间:"+time)
```

对象

全局对象：RegExp

- 什么是RegExp
- RegExp 是正则表达式的缩写。
- 当您检索某个文本时，可以使用一种模式来描述要检索的内容。RegExp 就是这种模式。
- 简单的模式可以是一个单独的字符。
- 更复杂的模式包括了更多的字符，并可用于解析、格式检查、替换等等。
- 您可以规定字符串中的检索位置，以及要检索的字符类型，等等。

对象

全局对象：RegExp

- 正则表达式的基本结构
- 一个正则表达式就是由普通字符（例如：字符 a~z）及特殊字符（称为元字符 ）组成的文字模式。该模式描述在查找文字主体时待匹配的一个或多个字符串。正则表达式作为一个模板，将某个字符模式与所搜索的字符串进行匹配。
- 语法：
- /匹配对象的模式/
- 位于“/”定界符之间的部分就是将在目标对象中进行匹配的模式。
- 例如：在字符串abcde中查找匹配模式bcd，代码如下：
- /bcd/

对象

全局对象：RegExp

- 正则表达式的语法规则
- 正则表达式的语法主要是对各个元字符功能的描述。元字符从功能上大致分为限定符、选择匹配符、分组组合符、反向引用符、特殊字符、字符匹配符和定位符。

特别字符	描述
$	匹配输入字符串的结尾位置。如果设置了 RegExp 对象的 Multiline 属性，则 $ 也匹配 '\n' 或 '\r'。要匹配 $ 字符本身，请使用 \$。
( )	标记一个子表达式的开始和结束位置。子表达式可以获取供以后使用。要匹配这些字符，请使用 \(和 \)。
*	匹配前面的子表达式零次或多次。要匹配 * 字符，请使用 *。
+	匹配前面的子表达式一次或多次。要匹配 + 字符，请使用 \+。
.	匹配除换行符 \n 之外的任何单字符。要匹配 . ，请使用 \. 。
[	标记一个中括号表达式的开始。要匹配 [，请使用 \[。
?	匹配前面的子表达式零次或一次，或指明一个非贪婪限定符。要匹配 ? 字符，请使用 \?。
\	将下一个字符标记为或特殊字符、或原义字符、或向后引用、或八进制转义符。例如， 'n' 匹配字符 'n'。'\n' 匹配换行符。序列 '\\' 匹配 "\" ，而 '\(' 则匹配 "("。
^	匹配输入字符串的开始位置，除非在方括号表达式中使用，此时它表示不接受该字符集合。要匹配 ^ 字符本身，请使用 \^。
{	标记限定符表达式的开始。要匹配 { ，请使用 \{。
|	指明两项之间的一个选择。要匹配 | ，请使用 \\|。

对象

全局对象：RegExp

- 正则表达式的语法规则
- 正则表达式的语法主要是对各个元字符功能的描述。元字符从功能上大致分为限定符、选择匹配符、分组组合符、反向引用符、特殊字符、字符匹配符和定位符。

限定符

限定符用来指定正则表达式的一个给定组件必须要出现多少次才能满足匹配。有 `*` 或 `+` 或 `?` 或 `{n}` 或 `{n,}` 或 `{n,m}` 共6种。

正则表达式的限定符有：

字符	描述
*	匹配前面的子表达式零次或多次。例如，zo* 能匹配 "z" 以及 "zoo"。* 等价于{0,}。
+	匹配前面的子表达式一次或多次。例如，'zo+' 能匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于{1,}。
?	匹配前面的子表达式零次或一次。例如，"do(es)?" 可以匹配 "do" 或 "does" 中的"do" 。? 等价于{0,1}。
{n}	n 是一个非负整数。匹配确定的 n 次。例如，'o{2}' 不能匹配 "Bob" 中的 'o'，但是能匹配 "food" 中的两个 o。
{n,}	n 是一个非负整数。至少匹配n 次。例如，'o{2,}' 不能匹配 "Bob" 中的 'o'，但能匹配 "foooood" 中的所有 o。'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o*'。
{n,m}	m 和 n 均为非负整数，其中n <= m。最少匹配 n 次且最多匹配 m 次。例如，"o{1,3}" 将匹配 "fooooood" 中的前三个 o。'o{0,1}' 等价于 'o?'。请注意在逗号和两个数之间不能有空格。

对象

- 定义RegExp
- RegExp 对象用于存储检索模式。
- 通过 new 关键词来定义 RegExp 对象。以下代码定义了名为 patt1 的 RegExp 对象，其模式是 "e"：

```
var patt1=new RegExp("e");
```

当您使用该 RegExp 对象在一个字符串中检索时，将寻找的是字符 "e"。

- RegExp 对象的方法

RegExp 对象的方法主要有test()、exec()、match()、search()和replace () 方法。

对象

全局对象：RegExp

- test()
- test() 方法检索字符串中的指定值。返回值是 true 或 false。
- 例如:
- <html>
- <body>

```
<script type="text/javascript">
var patt1=new RegExp("e");
document.write(patt1.test("The best things in life are free"));
</script>
</body>
```

- </html>
- 测试结果为true。

对象

全局对象：RegExp

- exec() 方法检索字符串中的指定值。返回值是被找到的值。如果没有发现匹配，则返回 null。
- 例如:
- <html>
- <body>

```
 <script type="text/javascript">
 var patt1=new RegExp("e");
 document.write(patt1.exec("The best things in life are free"));
 </script>
</body>
</html>
```

测试结果为: e

对象

全局对象：RegExp

- exec() 方法检索字符串中的指定值。返回值是被找到的值。如果没有发现匹配，则返回 null。添加一个参数g，则在使用 "g" 参数时，exec() 的工作原理如下：

找到第一个 "e"，并存储其位置

如果再次运行 exec()，则从存储的位置开始检索，并找到下一个 "e"，并存储其位置

- 例如：
- <html>
- <body>
- <script type="text/javascript">
- var patt1=new RegExp("e","g");
- do
- {
- result=patt1.exec("The best things in life are free");
- document.write(result);
- }
- while (result!=null)
- </script>
- </body>
- </html>测试结果为：eeeeeenull

对象

全局对象：RegExp

- match()
- 使用正则表达式模式对字符串执行查找，并将包含查找的结果作为数组返回。
- 例如:
- <html>
- <body>
- <script type="text/javascript">
- Function MatchDemo(){
- var r,re;
- var s=“I like JavaScript!”;
- re=/JavaScript/i;
- r=s.match(re);
- return(r);}
- Document.write(MatchDemo();)
- </script>
- </body>
- </html></html>测试结果为: JavaScript

对象

全局对象：RegExp

- 案例13

```
 <script language="javascript">
function checkeNO(NO){
    var str=NO;
     //在JavaScript中，正则表达式只能使用"/"开头和结束，不能使用双引号
    var Expression=/^\d{17}[\d|X]|^\d{15}$/;
    var objExp=new RegExp(Expression);
    if(objExp.test(str)==true){
            return true;
    }else{
            return false;
    }
}
</script>
```

想一想

- 1、使用Date对象的方法实现中文格式的日期和时间输出，日期的格式设置为“X年X月X日”，时间格式为“X时X分X秒”。
- 2、使用正则表达式验证邮箱的格式是否正确？