高等学校“十三五”规划教材

JavaScript 程序设计基础与实验指导

主编　李新荣

西安电子科技大学出版社
http://www.xduph.com

高等学校“十三五”规划教材

JavaScript程序设计基础与实验指导

主编　李新荣

西安电子科技大学出版社

★ 内容简介 ★

本书结合大量实例，详细介绍了 JavaScript 语法及内置对象。全书共分 8 章：第 1 章为 JavaScript 前端开发基础；第 2 章为 JavaScript 基础；第 3 章为函数及对象；第 4 章为数组对象；第 5 章为字符串对象；第 6 章为数学对象与日期对象；第 7 章为正则表达式；第 8 章为综合实验。本书理论与实验相结合，各章除理论知识外，均包含若干个实验，每个实验都有详细的分析和清晰的实现过程，便于读者在实践中逐步掌握 JavaScript 编程技术。

本书可作为大中专院校及培训学校计算机及相关专业的理论和实验实训教材，并可供 Web 前端开发人员参考。

图书在版编目(CIP)数据

JavaScript 程序设计基础与实验指导/李新荣主编. —西安：西安电子科技大学出版社，2017.9
ISBN 978-7-5606-4686-2

Ⅰ. ① J… Ⅱ. ① 李… Ⅲ. ① JAVA 语言—程序设计 Ⅳ. ① TP312.8

中国版本图书馆 CIP 数据核字(2017)第 217119 号

策　　划　陈婷
责任编辑　王静
出版发行　西安电子科技大学出版社(西安市太白南路 2 号)
电　　话　(029)88242885　88201467　　邮　　编　710071
网　　址　www.xduph.com　　电子邮箱　xdupfxb001@163.com
经　　销　新华书店
印刷单位　陕西华沐印刷科技有限责任公司
版　　次　2017 年 9 月第 1 版　2017 年 9 月第 1 次印刷
开　　本　787 毫米×1092 毫米　1/16　印 张 11.75
字　　数　274 千字
印　　数　1～3000 册
定　　价　23.00 元

ISBN 978 - 7 - 5606 - 4686 - 2/TP

XDUP 4978001-1

前　言

JavaScript是一种脚本语言，它通过提供动态的、个性化的交互式内容，来增强静态Web应用程序的功能。各种主流浏览器都支持JavaScript，所以JavaScript实际上是Web开发的首选脚本语言。

本书结合大量实例，详细介绍了JavaScript语法及内置对象。全书内容分8章：第1章为JavaScript前端开发基础；第2章为JavaScript基础；第3章为函数及对象；第4章为数组对象；第5章为字符串对象；第6章为数学对象与日期对象；第7章为正则表达式；第8章为综合实验。本书理论与实验相结合，实验部分介绍如何创建功能强大的Web应用程序。读者不必为没有编程经验而担心，本书将循序渐进地介绍所有相关知识。

各章内容可划分为三部分：第一部分对知识点进行讲述；第二部分是基础练习；第三部分是实践。在实践部分设有若干个实验，每个实验都给出了详细的分析和清晰的实现过程，包括实验目标(描述实验达到的目的)、实验内容及要求(提出实验的要求)、实验分析(包括结构分析和JavaScript算法分析)、实验步骤(描述实现的步骤)、总结与拓展(归纳案例实现的技巧，提出一些案例扩展)。

在学习本书时，首先要做到对知识点的透彻理解，而后再通过基础练习来巩固理论知识，最后再动手实践。

本书条理清晰，实用性和操作性强，可作为大中专院校及培训学校计算机及相关专业的理论和实验实训教材，并可供Web前端开发人员参考。

在本书编写过程中，感谢顾兴胜、周鑫鑫、周超、卢春丽、何鑫等同学为实验案例提供素材，感谢顾兴胜、刘红莲、李玲、朱任婷等同学参与本书的初稿试读和校对工作，他们站在初学者的角度对教材提出了许多宝贵的修改意见。

尽管编者已尽了最大的努力，但书中难免会有不妥之处，欢迎各界专家和读者朋友来信提出宝贵意见，我们将不胜感激。在阅读本书时，如发现任何问题或有不认同之处可以通过电子邮件与我们取得联系，邮件发送至 gdzyjsjx@126.com。

编　者

2017 年 4 月

目　录

第 1 章

JavaScript 前端开发基础

1.1　客户端的 JavaScript

1.“客户端”的程序

JavaScript 包含服务器端应用和客户端应用两个方面，本书主要介绍客户端应用的相关内容。在用户浏览器上运行的 JavaScript 程序，称为“客户端”的程序，用于根据用户的操作改变网页的画面或者进行动画处理等。

2. JavaScript 是一种面向对象的语言

(1) 对象。对象(object)就是某种东西。在现实中，看到的任何一个实物都称为对象，如课本、黑板等。对于 JavaScript，它处理的对象都在 Web 浏览器中，例如网页中的标题、段落、文本框、按钮等网页元素都是对象。

(2) 属性。对象具有属性，属性用来描述对象的状态和特征，例如按钮有 name、value 等属性。引用对象的属性，可以用“对象.属性名”引用。

(3) 方法。对象可以做的事情称为方法，JavaScript 对象也有方法，例如按钮可以单击。

3. window 对象

JavaScript 的“客户端”程序能在浏览器中运行，正是浏览器本身提供了这种脚本化的能力。window 对象是对浏览器当前窗口的引用。window 对象在客户端 JavaScript 中扮演着核心角色，它是客户端 JavaScript 程序的全局对象，并且可以用标识符 window 来引用它。window 对象定义了一些方法，比如 alert()方法，主要用于弹出警示对话框，通常用于对用户进行提示。window 对象还定义了一些属性，比如 document 属性。

4. document 对象

每个 window 对象有一个 document 属性引用 document 对象。document 对象表示当前窗口的网页内容(HTML 文档)，可以用来访问网页中的所有元素。document 对象定义了一些属性和方法，比如 document.getElementById()方法，即通过 id 获取页面的元素。

5. 元素对象

JavaScript 程序语言是一种基本对象的程序设计语言，它把网页上的元素如标题、段落、文本框、按钮、图像等，都作为“对象”处理。网页中的各元素之间的关系，描述为“对象”的层次结构关系，这种关系称为“文档对象模型 DOM”。

6. JavaScript 语言在 Web 开发中的作用

在 HTML 中，只是将文档的结构和内容组织起来，通过 CSS 进行渲染、布局等工作，而与用户交互的部分则由 JavaScript 来控制。JavaScript 可以用来修改 HTML 文档与 CSS 样式规则，向 HTML 文档中添加、插入及移除内容，增加或移除属性，修改样式。整个文档内容尽在 JavaScript 的控制之下。

7. JavaScript 处理用户交互事件

事件(event)是用户在访问页面时执行的操作。JavaScript 用事件处理程序来处理事件。用户在页面上的操作会触发脚本中的事件处理程序。如用户单击一个按钮，onclick 事件处理程序会执行，完成分配给它的任务。

8. 函数

函数(function)也可以叫作方法，是将一些代码组织在一起，形成一个用于完成某个具体功能的代码块，在需要时可以进行重复调用。

1.2 JavaScript 示例

1. JavaScript 程序编辑软件

编辑 JavaScript 程序可以用任何一种文本编辑器，如 EditPlus、Adobe Dreamweaver 等。下面示例用 Adobe Dreamweaver 编辑代码。

2. 编写 JavaScript 示例代码

打开 Adobe Dreamweaver，新建 HTML 文档，在代码视图中编辑如下程序代码。

示例 1.1　代码清单：

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>示例</title>
<style>
    div { width:200px; height:200px; background:red; display:none;}
</style>
</head>
<body>
    <input id="show_btn" type="button" value="显示" />
    <input id="hide_btn" type="button" value="隐藏" />
    <div id="div1"></div>
</body>
```

```
<script>
var oBtn1 = document.getElementById('show_btn');      //获取到页面上“显示”按钮对象
var oBtn2 = document.getElementById('hide_btn');      //获取到页面上“隐藏”按钮对象
var oDiv = document.getElementById('div1');           //获取到页面上 div 盒子对象
oBtn1.onclick = function (){                          //给“显示”按钮对象添加“单击”事件
    oDiv.style.display = 'block';                     //设置 div 盒子为显示
};
oBtn2.onclick = function (){                          //给“隐藏”按钮对象添加“单击”事件
    oDiv.style.display = 'none';                      //设置 div 盒子为隐藏
};
</script>
</html>
```

代码录入后，保存文件名为“ch1-01.html”。

3．在浏览器中运行

建议选择 Google Chrome 或 Mozilla Firefox 浏览器。这里在 Google Chrome 浏览器中观察“ch1-01.html”的运行效果。当用鼠标单击网页上的“显示”按钮时，显示一个红色的矩形，效果如图 1-1 所示。

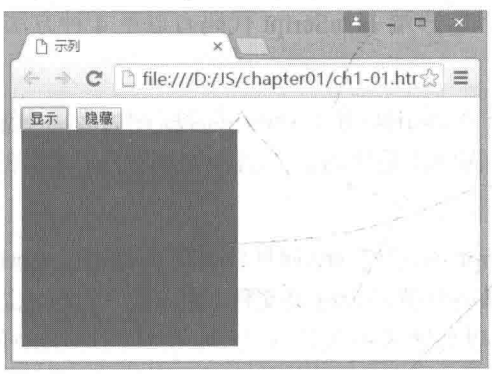

图 1-1　鼠标单击“显示”按钮时的网页效果

当用鼠标单击网页上的“隐藏”按钮时，红色的矩形消失，效果如图 1-2 所示。

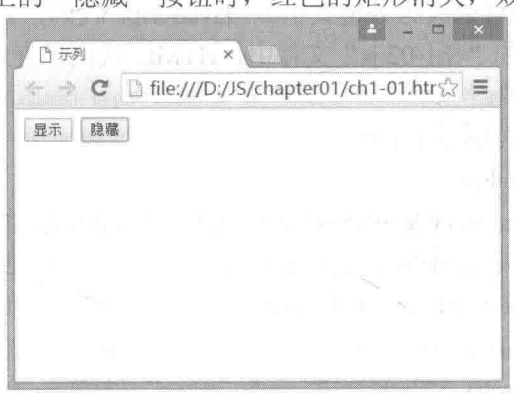

图 1-2　鼠标单击“隐藏”按钮时的网页效果

4．示例说明

(1) <script>标签。示例代码中，使用了<script>标签，在 HTML 文档中，JavaScript 的程序内容必须置于<script></script>标签对中，当浏览器读取到<script>标签时，就解释执行其中的脚本。其基本语法格式如下：

```
<script type="text/javascript">
    // 此处为 JavaScript 代码
</script>
```

(2) 示例 JavaScript 的程序的功能。此示例的 JavaScript 程序实现了对网页元素对象的处理，具体为：使用 document 对象的 getElementById()方法，通过网页元素 id 号获取到页面上的三个元素对象；通过用户单击网页上“按钮”这一交互行为，由 JavaScript 来控制 元素的显示属性值为“显示”还是“隐藏”。

(3) JavaScript 代码详解。JavaScript 代码详解，见程序清单中各语句的注解部分。代码清单中“//”后面的内容为注解。

1.3　在 HTML 文档中引入 JavaScript 方法

在 HTML 文档中引入客户端 JavaScript 代码有以下 4 种方法。

1．内嵌式

内嵌式是指代码包含在<script>和</script>标签对中，然后根据需要嵌入在 HTML 文档适当的位置，如示例 1.1 程序就是用内嵌式引入 JavaScript 代码的。

2．外链式

外链式是指通过<script>标记的 src 属性链接外部的 JavaScript 脚本文件。当脚本代码比较复杂或者同一段代码需要被多个网页文件使用时，可以将这些脚本代码放置在一个扩展名为.js 的文件中，然后以外链式引入该 js 文件。在 Web 页面中使用外链式引入 JavaScript 文件的基本语法格式如下：

```
<script type="text/javascript" src="JS 文件的路径"></script>
```

如示例 1.1 程序也可以改为外链式引入 JavaScript 代码：把 JavaScript 代码保存在“chapter01”文件夹下的“ch1-02.js”文件中，HTML 代码保存在“chapter01”文件夹下的“ch1-02.html”文件中。

“ch1-02.js”文件中的代码清单：

```
// JavaScript Document
var oBtn1 = document.getElementById('show_btn');      //获取到页面上“显示”按钮对象
var oBtn2 = document.getElementById('hide_btn');      //获取到页面上“隐藏”按钮对象
var oDiv = document.getElementById('div1');           //获取到页面上 div 盒子对象
oBtn1.onclick = function (){                          //给“显示”按钮对象添加“单击”事件
    oDiv.style.display = 'block';                     //设置 div 盒子显示
};
```

```
oBtn2.onclick = function (){                    //给“隐藏”按钮对象添加“单击”事件
    oDiv.style.display = 'none';                //设置 div 盒子隐藏
};
```

“ch1-02.html”文件中的代码清单：

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>示例</title>
<style>
    div { width:200px; height:200px; background:red; display:none;}
</style>
</head>
<body>
    <input id="show_btn" type="button" value="显示" />
    <input id="hide_btn" type="button" value="隐藏" />
    <div id="div1"></div>
<scrip type="text/javascript" src="ch1-02.js"></script>
</body>
</html>
```

3. 通过 HTML 文档事件处理程序引入 JavaScript 代码

根据用户的交互需求，可以给 HTML 文档中设定不同的事件处理器，通常是设置某 HTML 元素的属性来引用一个脚本(可以是一个简单的动作或者函数)，属性一般以 on 开头，如移动鼠标 onmousemove、单击鼠标 onclick 等属性。

示例 1.1 程序也可以改为通过 HTML 文档事件处理程序引入 JavaScript，修改后另存为文件名“ch1-03.html”。

“ch1-03.html”文件中的代码清单：

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>示例</title>
<style>
    div { width:200px; height:200px; background:red; display:none;}
</style>
</head>
<body>
    <input id="show_btn" type="button" value="显示"
```

```
        onclick="document.getElementById('div1').style.display='block';" />
        <input id="hide_btn" type="button" value="隐藏"
        onclick="document.getElementById('div1').style.display='none';"/>
        <div id="div1"></div>
    </body>
    </html>
```

4. 通过 JavaScript 伪 URL 地址引入 JavaScript 代码

用户还可以通过 JavaScript 伪 URL 地址调用语句来引入 JavaScript 脚本代码。伪 URL 地址的一般格式以“javascript:”开始，后面紧跟要执行的操作。例如如下网页文件中引入 JavaScript 脚本代码的方法。

“ch1-04.html”文件中的代码清单：

```
<!DOCTYPE HTML>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>示例</title>
</head>
<body>
    <a href="javascript:alert('hello')">Hello World!</a>
</body>
</html>
```

1.4 调试 JavaScript 程序

程序错误类型分为语法错误和逻辑错误两种，调试指的就是在代码中排查错误的过程。开发过程中调试是无法避免的事件，学会调试是程序设计人员的一个关键技能。JavaScript 本身提供的调试工具为 console 对象，浏览器提供了 JavaScript 程序调试工具，下面以 Chrome 浏览器的调试工具为例。调试示例“ch1-05.html”程序代码如下：

“ch1-05.html”文件中的代码清单：

```
<!DOCTYPE HTML>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>调试</title>
</head>
<body>
    <input id="count" type="text" />
```

```
    <input id="add" type="button" value="加 1"/>
    <script>
        var count=0;
        var oBtn=document.getElementById("add");
        var oCount=document.getElementById("count");
        oBtn.onclick=function(){
            count++;
            oCount.value=count;
        };
    </script>
</body>
</html>
```

1．Chrome 浏览器的调试工具

在 Chrome 浏览器中打开“ch1-05.html”网页，接着在页面上单击鼠标右键，在弹出的快捷菜单中选择“检查(N)”菜单项。打开开发者工具，如图 1-3 所示，开发者工具包含了 Elements 面板、Console 面板、Sources 面板、Network 面板、Timeline 面板、Profiles 面板、Resources 面板、Security 面板、Audits 面板。调试 JavaScript 程序在 Sources 面板，点击“Sources”标签后，进入 Sources 面板，如图 1-3 所示(打开开发者工具快捷方式：Ctrl+Shift+I(或者按 Ctrl+Shift+J 直接打开控制台)，或者直接按 F12)。

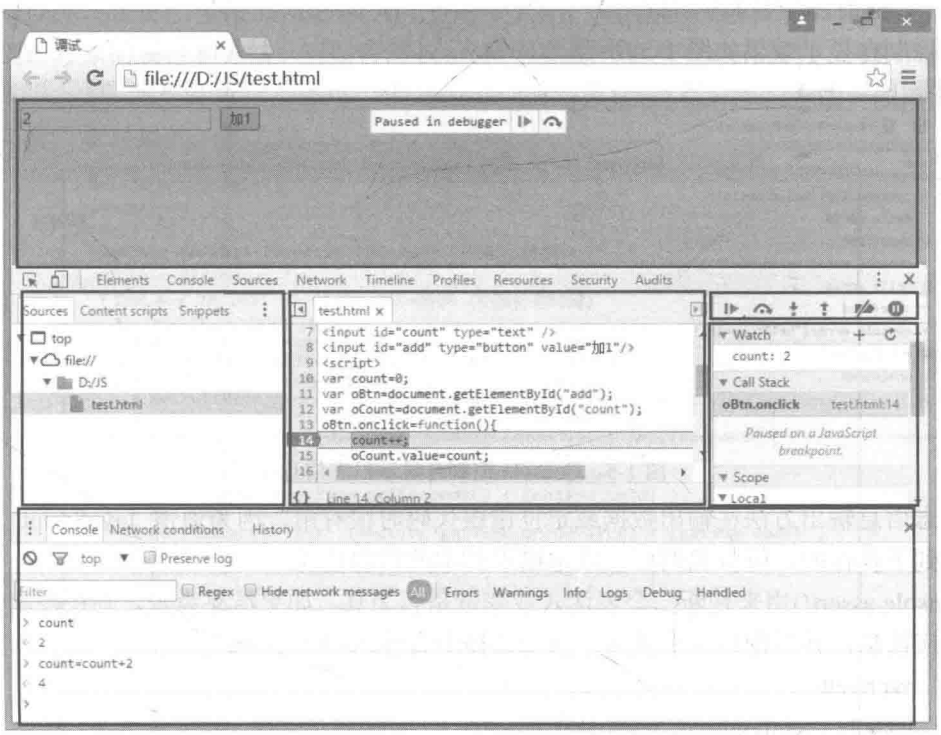

图 1-3　Chrome 浏览器开发者工具

窗口的最上方是网页内容显示区域，中间是 Sources 面板，窗口的最下方是 Console 控制台面板。

(1) Console 控制台面板。

① 如果 JavaScript 代码有语法错误，则控制台窗口能够显示当前页面中的 JavaScript 错误以及警告，并提示出错的文件和行号，方便调试，如图 1-4 所示。

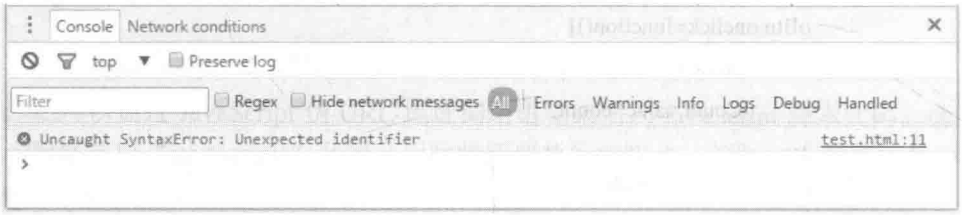

图 1-4　Console 控制台显示语法错误信息

② JavaScript 本身提供的调试工具 console 对象，其输出结果也是在浏览器控制台显示。可以通过 JavaScript 代码或者在控件台的命令行调用 console 对象的日志信息输出方法。console 对象的日志信息输出方法有 console.log()、console.warn() 和 console.error()，这些方法可以将日志信息输出到控制台。console.log()显示一般的基本日志信息，console.warn() 显示带有黄色小图标的警告信息，console.error()显示带有红色小图标的错误信息。

在控制台输入命令：

```
console.log('hello world')
console.warn('hello world')
console.error('hello world')
```

在控制台显示效果如图 1-5 所示。

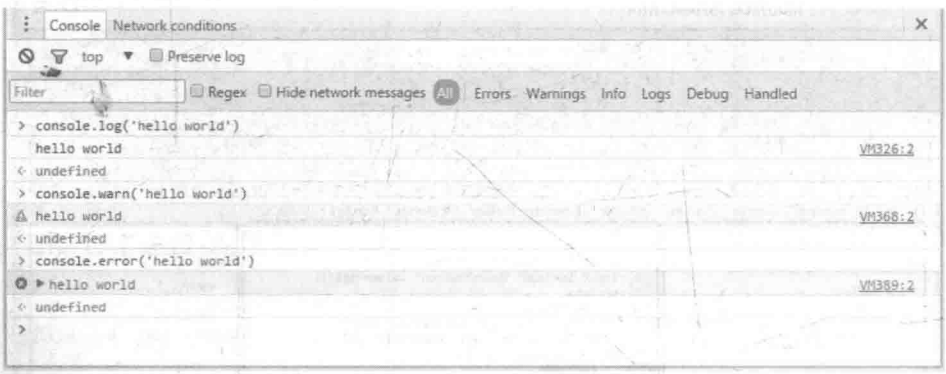

图 1-5　Console 控制台显示日志信息

日志信息输出方法在输出数据或定位错误代码时很有用，因为通常 JavaScript 文件都是从上到下执行的，所以我们可以锁定错误脚本的精确位置。

console.assert()用来判断一个表达式或变量是否为真。如果结果为否，则在控制台输出一条相应信息，并且抛出一个异常。例如在控制台输入如下命令：

```
var result=1;
console.assert(result==1, 'result 当前值为 1')
console.assert(result>1, 'result 当前值为 1')
```

控制台的显示效果如图 1-6 所示。

图 1-6　Console 控制台显示 assert 信息

③ 在控制台输入 JavaScript 表达式，按 Enter 即可得到表达式的值，在控制台输入命令时，会弹出相应的智能提示框，用户可以用 Tab 自动完成当前的建议项。在 Console 控制台中可以查看当前脚本的变量值，在控制台输入变量名，然后按回车键，变量值就会显示出来，可以改变当前变量的值。如图 1-3 控制台面板所示，在控制台输入 count 变量名，然后按回车键就能得到 count 变量名的值，输入 count=count+2，则还可以更改 count 变量的值。

④ 调试工具 console 对象，除了查看错误信息、打印调试信息(console.log())、写一些测试脚本以外，还可以当作 Javascript API 查看工具用，直接在控制台输入对象名后回车，可查看到该对象的属性和方法。例如查看 console 对象都有哪些方法和属性，可以直接在控制台输入 console 回车并执行；还可以用 console.dir(对象)来查看对象的方法。图 1-7 所示给出了查看日期对象的一些方法。

```
> var d = new Date()
  undefined
> d
  Sun Aug 28 2011 21:04:18 GMT+0800 (中国标准时间)
> console.dir(d)
  ▼Sun Aug 28 2011 21:04:18 GMT+0800 (中国标准时间)
    ▼__proto__: Invalid Date
      ▶constructor: function Date() { [native code] }
      ▶getDate: function getDate() { [native code] }
      ▶getDay: function getDay() { [native code] }
      ▶getFullYear: function getFullYear() { [native code] }
      ▶getHours: function getHours() { [native code] }
      ▶getMilliseconds: function getMilliseconds() { [native code] }
      ▶getMinutes: function getMinutes() { [native code] }
      ▶getMonth: function getMonth() { [native code] }
      ▶getSeconds: function getSeconds() { [native code] }
      ▶getTime: function getTime() { [native code] }
      ▶getTimezoneOffset: function getTimezoneOffset() { [native code] }
      ▶getUTCDate: function getUTCDate() { [native code] }
      ▶getUTCDay: function getUTCDay() { [native code] }
      ▶getUTCFullYear: function getUTCFullYear() { [native code] }
      ▶getUTCHours: function getUTCHours() { [native code] }
      ▶getUTCMilliseconds: function getUTCMilliseconds() { [native code] }
      ▶getUTCMinutes: function getUTCMinutes() { [native code] }
      ▶getUTCMonth: function getUTCMonth() { [native code] }
      ▶getUTCSeconds: function getUTCSeconds() { [native code] }
      ▶getYear: function getYear() { [native code] }
      ▶setDate: function setDate() { [native code] }
      ▶setFullYear: function setFullYear() { [native code] }
      ▶setHours: function setHours() { [native code] }
```

图 1-7　查看日期对象的方法