Web前端开发技术与实践

第14章：初始JavaScript

阮晓龙

13938213680 / rxl@hactcm.edu.cn
http://web.book.51xueweb.cn

河南中医学院管理信息工程学科
河南中医学院网络信息中心

2015.9

本章主要内容

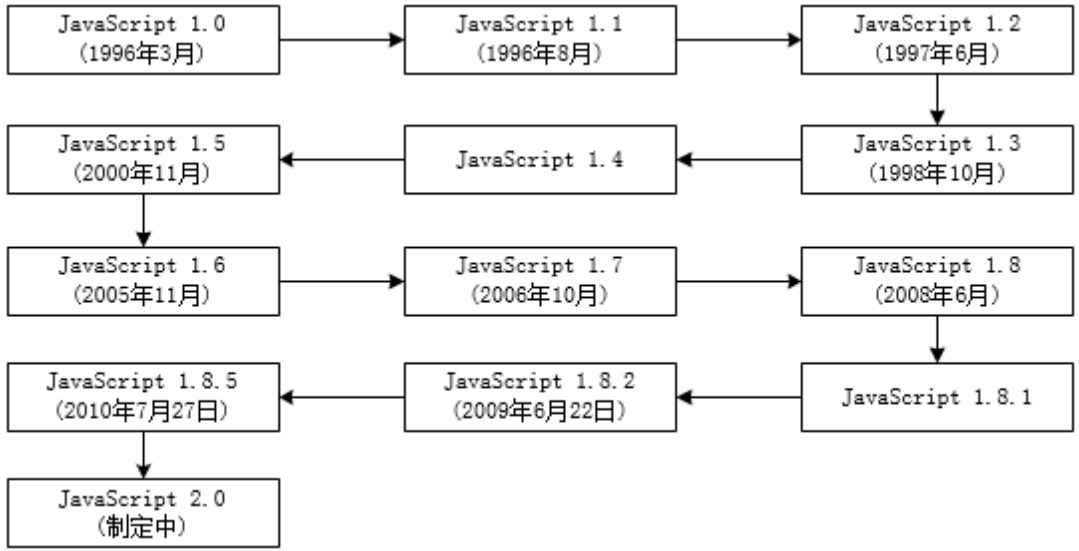
- JavaScript概述
- JavaScript语法
- DOM
- 案例：使用JavaScript进行表单验证
- 案例：使用JavaScript实现规定时间内答题效果

1.JavaScript概述

- JavaScript是一种为网站添加互动以及自定义行为的客户端脚本语言，因此通常只能通过Web浏览器去完成操作，而无法像普通意义上的程序那样独立运行。

1.JavaScript概述

- 发展历程
  - JavaScript与Java没有任何的关系，它由Netscape公司与Sun公司合作开发。JavaScript最开始的名字是LiveScript，因当时Java风靡一时以及当时正与Sun公司进行合作等因素，于是将LiveScript改为了JavaScript。JavaScript的第一个版本，出现在1996年推出的NetScape Navigator 2浏览器中。


JavaScript 1.0
(1996年3月)
JavaScript 1.1
(1996年8月)
JavaScript 1.2
(1997年6月)
JavaScript 1.3
(1998年10月)
JavaScript 1.4
JavaScript 1.5
(2000年11月)
JavaScript 1.6
(2005年11月)
JavaScript 1.7
(2006年10月)
JavaScript 1.8
(2008年6月)
JavaScript 1.8.1
JavaScript 1.8.2
(2009年6月22日)
JavaScript 1.8.5
(2010年7月27日)
JavaScript 2.0
(制定中)

1.JavaScript概述

- 主要特点
  - 解释性执行的脚本语言
    - JavaScript的语法基本结构形式与C、C++、Java十分类似，但是在使用之前，不需要先编译，而是在程序执行中被逐行的解释。
  - 简单弱类型脚本语言
    - JavaScript的简单性主要在于其基于Java基本语句和控制流之上的简单而紧凑的设计；其次在于其变量类型是采用弱类型，并未使用严格的数据类型。

1.JavaScript概述

- 相对安全的脚本语言
  - JavaScript作为一种安全性语言，不被允许访问本地硬盘，且不能将数据存入服务器，不允许对网络文档进行修改和删除，只能通过浏览器实现信息浏览或动态交互，从而有效地防止数据的丢失或对系统的非法访问。
- 跨平台性的脚本语言
  - JavaScript依赖于浏览器本身，与操作环境无关，只要计算机能运行支持JavaScript的浏览器，就可正确执行，从而实现了跨平台的特性。

1.JavaScript概述

- 功能概述
  - 通常情况下，Web前端开发者使用JavaScript在给网页添加交互作用。网页的结构层是HTML；网页表现层由CSS构成；网页行为层由JavaScript组成。网页上的所有元素、属性和文本都能通过使用DOM（文本对象模型）的脚本来获得。
  - Web前端开发者可通过JavaScript来实现改变网页内容、CSS样式、对用户输入做出反馈等操作。

2.JavaScript语法

- 用JavaScript编写的代码必须通过HTML/XHTML文档才能执行。目前有两种方法可以调用JavaScript。
  - 方法一：将JavaScript代码放到文档<head>或<body>标签中的<script>标签之间。但最好的做法是将<script>标签放到HTML文档的最后，<body>结束标签之前。
  - 方法二：将JavaScript代码存为一个扩展名为.js的独立文件。典型的做法是在文档的<head>部分放置一个<sccript>标签，并把它的src属性指向该文件。

2.JavaScript语法

- 方法一：
  - 放在<head>标签中。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>第一个 JavaScript 编程</title>
<script>
        //JavaScript 代码
</script>
</head>
<body>
<div id="bodyContent" class="body-content">
</div>
</body>
</html>
```

2.JavaScript语法

- 方法二：
  - 放到HTML文档的最后，<body>结束标签之前

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>第一个 JavaScript 编程</title>
<script src="file.js"></script>
</head>
<body>
<div id="bodyContent" class="body-content">
</div>
</body>
</html>
```

2.JavaScript语法

- 需要主要的是：上述代码中<script>标签没有包含传统的type="text/javascript"属性，是因为在HTML5规范中，script属性默认是text/javascript，所以可以省略，但是在HTML4.01和XHTML1.0规范中，type属性是必须的。

2.JavaScript语法

- 语法结构
  - JavaScript程序使用Unicode字符集编写，它是一种区分大小写的语言，也就是说，在输入关键字、变量、函数名以及所有的标识符时，都必须采取一致的字母大小写格式。
  - JavaScript会忽略程序中记号之间的空格、制表符和换行符。因为可以在程序中任意使用空格、制表符和换行符，因此开发者可以采用整齐、一致的方式排版JavaScript代码，增加代码的可读性。
  - JavaScript中的简单语句后面通常都有分号（;）主要是为了分割语句。

2.JavaScript语法

- JavaScript和Java一样，它也支持C++、C型的注释。JavaScript会把处于“//”和一行结尾之间的任何文本都当做注释忽略掉。此外“/*”和“*/”之间的文本也会被当做注释，这个注释可以跨越多行，但是其中不能有嵌套的注释。

```
//这是一条单行注释
/*
这是一个多行注释
我是第二行注释内容
我是第三行注释内容
*/
/*这是一条注释*/      //这是另一条注释
```

2.JavaScript语法

- JavaScript的直接量，就是程序中直接显示出来的数据值。

```
12                  //这是整数直接量
1.2                 //这是浮点直接量
"Hello World"       //这是字符串直接量
'Hi'                //这是另一个字符串直接量
true                //这是布尔直接量
false               //这是另一个布尔直接量
null                //这是一个空对象
{x:1,y:3}           //对象直接量
[1,2,3,4]           //数组直接量
```

2.JavaScript语法

- 在JavaScript中标识符用来命名变量与函数名，或者用作JavaScript代码中某些循环的标签。JavaScript合法的标识符命名规则为：第一个字符必须为字母、下划线或美元符号（$），接下来的字符可以是字母、数字、下滑线或美元符号，数字不允许作为首字母出现。另外标识符不能和JavaScript中用于其它用途的关键字同名。同时需要注意的是，JavaScript中的保留字在JavaScript程序中不能被用作标识符。

2.JavaScript语法

- ECMAScript v3标准化的关键字如表所示。

表 14-01 保留的 JavaScript 关键字

break	do	If	switch	typeof
case	else	In	this	var
catch	false	instanceof	throw	void
continue	finally	New	true	while
default	for	Null	try	with
delete	function	return		

2.JavaScript语法

- ECMA扩展保留的关键字如表所示。

表 14-02 ECMA 扩展保留的关键字

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

2.JavaScript语法

- 此外还应该避免把JavaScript预定义的全局变量名或全局函数名用作标识符。避免使用的标识符如表所示。

表 14-03 要避免使用的其他标识符

arguments	encodeURI	Infinity	Object	String
Array	Error	isFinite	parseFloat	SyntaxError
Boolean	escape	isNaN	parseInt	TypeError
Date	eval	NaN	RangeError	undefined
decodeURI	EvalError	Number	ReferenceError	unescape
decodeURIComponent	Function	Math	RegExp	URIError

2.JavaScript语法

- 数据类型
  - JavaScript是一种弱类型语言，这意味着Web前端开发者可以在任何阶段改变变量的数据类型，而无需像强类型语言一样在声明变量的同时还必须同时声明变量的数据类型。
  - JavaScript的三种基本数据类型有：字符串、数值、布尔值。

2.JavaScript语法

- 字符串
  - 字符串由零个或多个字符构成。字符包括（但不局限于）字母、数字、标点符号和空格。字符串必须包在引号里面，单引号或双引号都可以。JavaScript可以随意的选用引号，但最好还是根据字符串所包含的字符来选择。即如果字符串包含双引号，就把整个字符串包含在引号里面；如果包含单引号就把整个字符放在双引号里面。

```
var mood="don't ask";
var mood='中国"飞人"勇夺金牌';
```

  - 如果一个字符串中既有单引号又有双引号，那么这种情况下需要把那个单引号或双引号看做一个普通字符，而不是这个字符串的结束标志，这种情况下需要对这个字符进行转义，在JavaScript中用反斜线对字符串进行转义

```
var height="It's about 5'10\" tall";
```

2.JavaScript语法

- 数值
  - 如果想给一个变量赋一个数值，不用限定它必须是一个整数。JavaScript允许使用带有小数点的数值，并且允许任意位小数点，这样的数称为浮点数，数值主要数据类型如下所示。

```
var num=33.25                    //这是一个浮点数
num=-88;                         //这是一个负数
num=-20.333                      //这是一个负数浮点数
```

2.JavaScript语法

- 布尔值
  - 布尔数据只有两个可选值：true或者false。
  - 布尔值不是字符串，不能将布尔值用引号括起来。布尔值的false与字符串值“false”是完全不相关的两码事。

```
var married=false;                    //变量 married 设置为布尔值 false
var married="false";                  //变量 married 设置为字符串“false”
```

2.JavaScript语法

- 变量
  - 变量通常指那些会发生变化的东西，把值存入变量的操作统称为赋值，在JavaScript中可以用下面的代码进行赋值。

```
mood="happy";
age=33;
```

  - 在JavaScript中程序可不必直接对变量赋值而无需事先声明。作出变量声明如下所示。

```
var mood;
var age;
var age,mood;                    //一次声明两个变量
```

  - 声明变量的同时，完成赋值如下所示。

```
var mood="happy";
var age=24;
var mood="happy",age=24;         //一次声明赋值两个变量
```

2.JavaScript语法

- 表达式与运算符
  - 表达式
    - 表达式是JavaScript的一个“短语”，JavaScript解释器可以计算表达式，从而生成一个值，最简单的表达式是直接量或变量名。
    - 直接量表达式的值就是这个直接量本身，变量表达式的值则是该变量所存放或引用的值。
    - 通过合并简单的表达式可以创建较为复杂的表达式，具体代码如下所示。

```
i+0.7
```

2.JavaScript语法

- 运算符
  - JavaScript中，加减乘除都是一种操作，这些算术操作中的每一种都必须借助于相应的操作符才能完成，操作符是JavaScript为完成各种操作而定义的一些符号。假设y=3，则：

表 14-04 JavaScript 算数运算符

运算符	描述	例子	结果
+	加	x=y+2	x=5
-	减	x=y-2	x=1
*	乘	x=y*2	x=6
/	除	x=y/2	x=1.5
%	求余数	x=y%2	x=1
++	累加	x=++y	x=4
--	递减	x=--y	x=2

2.JavaScript语法

- 赋值运算符用于给JavaScript变量赋值，假设x=6，y=3，则：

表 14-05 JavaScript 赋值运算符

运算符	例子	等价于	结果
=	x=y		x=3
+=	x+=y	x=x+y	x=9
-=	x-=y	x=x-y	x=3
*=	x*=y	x=x*y	x=18
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

- 加号（+）是一个比较特殊的操作符，它既可以用于数字，也可用于字符串，代码如下所示。

```
var message="I am feel "+"good";
```

2.JavaScript语法

- 流程控制语句
  - If语句
    - if语句是最常见的条件语句，基本语法如下所示。条件必须放在if后面的圆括号中。条件的求值结果永远是一个布尔值，即只能为true或false。花括号中的语句不管内容有多少条，只有在给定条件的求值结果为true的情况下才会执行。

```
if(condition){
    //执行语句内容
}
```

    - 例如：

```
if(1+1=3){
    alert("It's wrong");
}
```

2.JavaScript语法

- 流程控制语句
  - If语句
    - if语句是最常见的条件语句，基本语法如下所示。条件必须放在if后面的圆括号中。条件的求值结果永远是一个布尔值，即只能为true或false。花括号中的语句不管内容有多少条，只有在给定条件的求值结果为true的情况下才会执行。

```
if(condition){
    //执行语句内容
}
```

    - 例如：

```
if(1+1=3){
    alert("It's wrong");
}
```

2.JavaScript语法

- if语句的第二种形式引入了else从句，当给定条件的求值结果为false时，就会执行这个else从句，其基本语法结构如下所示：

```
if(condition){
    //执行语句内容
}else{
    //执行语句内容
}
```

- 例如：

```
if(1+1=3){
    alert("It's wrong");
}else{
    alert("It's right");
}
```

2.JavaScript语法

- switch语句
  - 一个if语句会在程序的执行流程中产生一个分支，但是当程序含有多个分支，并且所有的分支都依赖于一个变量的值时，多个if语句重复性的检测同一个变量的值将会产生资源浪费。而switch语句正是用来处理这种情况的，它比重复使用if语句要高效的多，其语法结构如下所示：

```
switch(expression){
    //执行代码内容
}
```

2.JavaScript语法

- 在执行代码内容中，不同的位置要使用case关键字后加一个值和一个冒号来标记。当执行一个switch语句时，它先计算expression的值，然后查找与这个值匹配的case标签，找到相应的case标签，就开始执行case标签后的代码块语句，如果没有相匹配的内容，就开始执行标签default后的语句，如果没有default标签，就跳过所有的代码块。

```
switch(type)
{
    case 0:
        alert("HTML");
        break;
    case 1:
        alert("CSS");
        break;
    case 2:
        alert("JavaScript");
        break;
}
```

2.JavaScript语法

- While循环
  - while循环语句与if语句十分相似，它们的语法几乎一样，其语法结构如下所示：

```
while(expression){
    //执行语句内容
}
```

  - 例如：

```
var count=1;
while(count<11){
    alert(count);
    count++;
}
```

  - while循环语句与if语句唯一的区别是：只要给定条件的求值结果是true，包含在花括号里的代码就将反复执行下去。

2.JavaScript语法

- 在某些场合，我们希望那些包含在循环语句内部的代码至少执行一次，这时我们便需要使用do循环了，其语法结构如下所示：

```
do{
    //执行语句内容
}while(condition)
```

- 例如：

```
var count=1;
do{
    alert(count);
    count++;
}while(count<11)
```

- do循环与while循环最大的区别便是：对循环控制条件的求值发生在每次循环结束之后。因此，即使循环控制条件的首次求值结果为false，花括号里的语句也至少会被执行一次.

2.JavaScript语法

- For循环
  - 在JavaScript中使用for循环来执行一些代码十分方便，它类似于while循环。事实上，for循环只是while循环的一种变体，而for循环不过是进一步改写为如下所示的紧凑格式而已，其语法结构如下所示：

```
for(initial condition;test condition;alter condition){
    //执行语句内容
}
```

  - 例如：

```
for(var count=1;count<11;count++){
    alert(count);
}
```

  - 用for循环来重复执行一些代码的好处是循环控制结构更加清晰。与循环有关的所有内容都包含在for语句的圆括号里面。

2.JavaScript语法

- for循环最常见的用途之一便是对某个数组里的全体元素进行遍历处理。这往往需要用到数组的array.length属性，这个属性表示给定数组里元素的个数，切记数组下标是从0开始的，下面的例子是指循环输出数组中的所有元素：

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

2.JavaScript语法

- for/in语句
  - 在JavaScript中关键字for有两种使用方式。我们前面已经讲过其在for循环中的使用情况，此外它还可以用于for/in语句，其语法结构如下所示：

```
for(variable in object){
    //执行语句内容
}
```

  - 例如：

```
var myArray=["BMW","Volvo","Saab","Ford"];
var i;
for (i in myArray){
    alert(myArray[i]);
}
```

2.JavaScript语法

- variable是指一个变量名，声明一个变量的var语句，数组的一个元素或者是对象的一个属性。object是一个对象名，或者是计算结果为对象的表达式。
- JavaScript的数组是一种特殊的对象，因此for/in循环可以像枚举对象属性一样枚举数组的下标。
- for/in循环并没有指定将对象的属性赋给循环变量的顺序。因为没有什么办法可以预先告知其赋值顺序，因此在不同的JavaScript版本或者实现中实现这一语句的行为可能有所不同。

2.JavaScript语法

- break语句
  - 在JavaScript中break语句会使运行的程序立刻退出包含在最内层的循环或者退出一个switch语句，其语法结构如下所示：

```
break;
```

  - 由于其用来退出循环或者switch语句，因此只有当它出现在这些语句当中时，这种形式的break语句才能被解析。
  - JavaScript允许关键字break后跟一个标签名，当break和标签一起使用时，它将跳转到这个带有标签的语句的尾部，或者禁止这个语句。该语句可以是任何用括号括起来的语句，它不一定是循环语句或者switch语句。

2.JavaScript语法

- continue语句
  - continue语句与break语句相似，不同的是它不是退出一个循环而是开始循环的一次新迭代，其语法结构如下所示：

```
continue;
```

  - continue语句只能在while语句、do/while语句、for语句或者for/in语句的循环体中使用，在其它地方使用将不会被解析。
  - 执行continue语句时，封闭循环的当前迭代就会被终止，开始执行下一次迭代，这对不同类型的循环语句来说含义是不同的：
  - 在while循环语句中，会再次检测循环开头的expression，如果值为true，将从头开始执行循环内容；

2.JavaScript语法

- 在do/while循环中，会跳到循环的底部，在顶部开始下一次循环之前会在此先检测循环条件；
- 在for循环中，先计算increment表达式，然后再检测test表达式以确定是否应该执行下一次迭代；
- 在for/in循环中，将以下一个赋给循环变量的属性名开始新的迭代。
- 在while循环和for循环中continue语句行为的不同之处在于，while循环是直接跳到循环条件处，而在for循环中则要先计算increment表达式，然后再跳转到循环条件处。

2.JavaScript语法

- throw语句
  - 所谓的异常通常就是指一个信号，说明发生了某种异常情况或错误。抛出一个异常，就是用信号通知发生了错误或异常情况。扑捉一个异常，就是处理它，即采取必要或适当的动作从异常恢复。在JavaScript中，当发生运行时错误或程序明确的使用throw语句时就会抛出异常。使用try/catch/finally语句可以捕获异常，这个我们将在下一节介绍。throw语句使用语法结构如下所示：

```
throw expression;
```

2.JavaScript语法

- expression的值可以是任何类型的，但是通常情况下它是一个Error对象或Error子类的一个实例。例如：

```
function faction(x){
    if(x<0){
        throw new Error("x is a wrong number");
    }
    for(var i=0;i<10;i++){
        x++;
    }
    return x;
}
```

- 在抛出异常时，JavaScript解释器会立即停止正常的程序执行，跳转到最近的异常处理器，如果抛出异常的代码块没有相关的catch从句，解释器将检查次高级的封闭代码块，看它是否有相关的异常处理器，以此类推，直到找到一个异常处理器为止。

2.JavaScript语法

- try/catch/finally语句
  - try/catch/finally语句是JavaScript的异常处理机制。该语句的try从句只定义异常需要被处理的代码块。catch从句跟随在try从句后面，它是try从句内的某个部分发生了异常调用的语句块。finally从句跟随在catch从句后，存放清除代码，无论try从句中发生了什么，该代码块都会被执行。虽然catch从句和finally从句都是可选的，但是try从句中至少应该有一个catch从句或finally从句。try、catch、finally从句都以大括号开头和结尾，这是必须的语法部分，即使从句只有一条语句，也不能省略大括号。

2.JavaScript语法

- try/catch/finally示例如下所示：

```
try{
    var num=1;
    var num2=faction(num1);
    alert(num2);
}catch(e){
    alert(e);
}finally{
    alert("end");
}
```

2.JavaScript语法

- 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();
```

2.JavaScript语法

- JavaScript内置了许多的函数，在前面多次用到的alert就是其中的一种，这个函数需要我们提供一个参数，它将弹出一个对话框来显示这个参数的值。
- 在定义函数时，可以为它声明多个参数，只要用逗号将其隔开就行。在函数内部，可以像使用普通变量那样使用它的任何一个参数。一个进行乘法运算的函数如下所示：

```
function multipy(num1,num2){
    var total=num1*num2;
    alert(total);
}
```

- 在定义函数的脚本后，可以使用如下所示的语法进行调用：

```
multipy(10,2);
```

2.JavaScript语法

- 然而很多语法调用这个函数只是为了得到最终结果而非在页面上展示出来，因此我们需要函数不仅能够（以参数的形式）接收数据，还能够返回数据。这时便需要用到return语句了，改造后的函数如下所示：

```
function multipy(num1,num2){
    var total=num1*num2;
    return total;
}
```

2.JavaScript语法

- 变量既可以是全局的，也可以是局部的，全部变量与局部变量的区别就在于其作用域。
- 全局变量可以在脚本的任何位置被引用，其作用域为整个脚本。
- 局部变量只存在于声明它的那个函数的内部，在那个函数的外部是无法引用它的，作用域仅为某个特定的函数。

2.JavaScript语法

- 通过var关键字明确的声明变量时，如果在函数中使用了var，那这个变量就会被视为一个局部变量，它只存在于这个函数的上下文中；反之，如果没有使用var，那这个变量就被视为一个全局变量，如果脚本里已存在一个与之同名的全局变量，这个函数将会改变全局变量的值。例如：

```
function square(num){
    total=num*num;
    return total;
}
var total=50;
var number=square(total);
alert(number);
```

2.JavaScript语法

- 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，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()
```

2.JavaScript语法

- 内建对象
  - 在JavaScript中内置了一些对象，比如前面用到的数组。当我们使用new关键字去初始化一个数组是，其实就是在创建一个Array对象的新实例，如下代码所示：

```
var beatles=new Array();
```

  - Array对象只是诸多JavaScript内建对象中的一种。其它的还包含Math对象、Date对象，它们分别提供了许多非常有用的方法供人们处理数值和日期值，比如，Math对象的round方法可以把十进制数值舍入为一个与之最接近的整数，其代码如下所示：

```
var num=7.561;
var number=Math.round(num);
alert(number);
```

2.JavaScript语法

- Date对象可以用来存储和检索与特地日期和时间有关的信息。在创建Date对象的新实例时，JavaScript解释器将自动的使用当前日期和时间对它进行初始化，其代码如下所示：

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

- Date对象提供了getDay()、getHours()、getMonth()等一系列方法，以供人们用来检索与特定日期有关的各种信息。

2.JavaScript语法

- 宿主对象
  - 除了内置对象以外，还可以在JavaScript脚本里使用一些已经预先定义好的其它对象。这些对象不是由JavaScript语言本身而是由它的运行环境提供的，在Web应用中，这个环境就是浏览器，由浏览器提供的预定义对象被称为宿主对象。
  - 宿主对象包括Form、Image和Element等。我们可以通过这些对象获得关于网页上表单、图像和各种表单元素等信息，其中最重要的一个宿主对象便是document对象。

3.DOM

- 文档对象模型简称DOM，DOM是一种HTML/XHTML页面的编程接口（API）。它提供了一种文档的结构化地图，还有一组方法，以便与所含元素交互。实际上，它是把我们的标记方式转换为JavaScript可以理解的格式。简单的说就是DOM就像页面上的所有元素的一个地图。我们可以使用它通过名字和元素来找到元素，然后添加、修改或删除元素及其内容。

3.DOM

- 在DOM中有三种方法能够获取元素节点，分别是通过元素ID、通过标签名称和通过类名称来进行获取。

3.DOM

- getElementById
  - DOM提供了一个名为getElementById的方法，这个方法将返回一个与哪个有着给定id属性值的元素节点对应的对象。
  - getElementById是document对象特有的函数。在脚本代码里，函数名的后面必须跟有一对圆括号，这对圆括号包含着函数的参数。getElementById方法只有一个参数：想要获得的那个元素的id属性的值，这个id属性值必须放在单引号或者双引号里，其使用方法如下所示：

```
document.getElementById("purchases");
```

3.DOM

- getElementsByTagName
  - getElementsByTagName方法返回一个对象数组，每个对象分别对应着文档里有着给定标签的一个元素。类似于getElementById，这个方法也是只有一个参数的函数，它的参数是标签的名字，其使用方法如下所示：

```
element.getElementsByTagName(tag);
```

3.DOM

- getElementsByClassName
  - HTML5 DOM中新增了一个方法：getElementsByClassName。这个方法能够使我们通过class属性中的类名来访问元素。不过由于这是一个新增方法，某些DOM实现中可能还未支持此方法的解析，其在Internet Explorer 5，6，7，8中无效，因此在使用时要注意其兼容性。
  - getElementsByClassName方法与getElementsByTagName方法相似，也只接受一个参数，就是类名，其使用方法如下所示：

```
element.getElementsByTagName(class);
```

3.DOM

- 增加元素
  - 如果我们需要向HTML中添加新元素，那么我们首先便需要创建该元素，然后向已存在的元素追加创建的新元素。
  - document.createElement()方法和document.createTextNode()方法分别用来创建新的Element节点和Text节点，而方法node.appendChild()、node.insertBefore()和node.replaceChild()可以用来将它们添加到一个文档，其具体实现方法如下所示：
    - 首先创建一个新的元素，比如<p>，其代码如下所示：

```
var para=document.createElement("p");
```

3.DOM

- 首先创建一个新的元素，比如<p>，其代码如下所示：

```
var para=document.createElement("p");
```

- 如果需要向<p>元素中添加文本内容，必须先创建一个文本节点，如以下代码所示：

```
var node=document.createTextNode("这是创建的新段落。");
```

- 然后将该文本节点追加到刚开始创建的<p>元素中，代码如下所示：

```
para.appendChild(node);
```

- 最后必须向一个已有的元素追加这个新建的元素，其实现代码如下所示：

```
var element=document.getElementById("div1");
element.appendChild(para);
```

3.DOM

- 修改元素
  - 修改元素内容
    - 修改元素内容的最简单的方法便是使用innerHTML属性，使用该属性可以对元素的内容重新赋值，从而达到修改元素内容的效果，其使用方法如下所示：

```
document.getElementById(id).innerHTML=new HTML;
```

3.DOM

- 改变元素属性
  - 在得到需要的元素以后，我们就可以设法获取它的各个属性，getAttribute方法就是专门用来获取元素属性的，相应的我们也可以使用setAttribute方法来更改元素节点的值。
  - getAttribute是一个函数，它只有一个参数即：查询的属性的名称，其使用方法如下所示：

```
object.getAttribute(attribute);
```

  - setAttribute()方法是用来进行设置属性的，它允许我们对属性节点的值做出修改，与getAttribute方法一样，它也是只能用于元素节点，其使用方法如下所示：

```
object.setAttribute(attribute,value);
```

3.DOM

- 删除元素
  - 如果需要在HTML中删除元素，那么我们首先便需要获得该元素，然后得到该元素的父元素，最后通过removeChild方法删除该元素，其实现流程如下所示：
  - 获得该元素，比如要获得id属性值为div1的元素，其代码如下所示：

```
var child=document.getElementById("p1");
```

  - 获得该元素的父元素，代码如下所示：

```
var parent=document.getElementById("div1");
```

  - 从父元素中删除该元素

```
parent.removeChild(child);
```

4.案例：使用JavaScript进行表单验证

- 使用JavaScript代码验证输入值是否为空、是否为整数、是否为正确时间等信息，点击提交后在页面上显示提示信息。在进行精确验证时用到JavaScript的正则表达式，准确匹配输入值格式。

现场演示：

- 案例14-01：使用JavaScript进行表单验证

5.案例：规定时间内答题效果的实现

- 实现一个在线答题页面，用户可以直接在网页中进行答题，如果超出答题时间，将直接跳过该题目，所有问题回答完毕后给出一个提示信息的效果。

现场演示：

- 案例14-02：使用JavaScript实现规定时间内答题效果

Thanks.