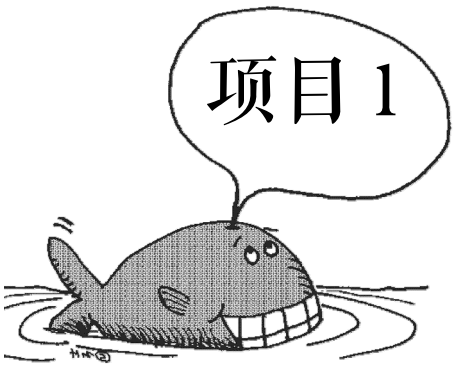

对话框和页面输出——初步体验

项目情境

有两张网页，每张网页上都有一个“问候”按钮。在第一张网页上单击该按钮，弹出对话框“张三向您问好!”，单击对话框中的“确定”按钮后，在该网页上打印“张三欢迎您光临本站!”。在第二张网页上做同样的操作，显示结果相同，但信息中把“张三”显示为“李四”。项目要求代码的复用性和可维护性良好。

学习目标

- 熟悉 JavaScript 内嵌在 HTML 中的书写方法。
- 熟悉 JavaScript 的执行时机。
- 初步了解事件触发的概念。
- 初步了解函数和函数调用的方法。
- 熟悉 JavaScript 文件的编写和引入的方法。

任务 1　弹出对话框

提出任务

打开页面，立即弹出一个问候对话框。

知识预备

（1）在 HTML 中，任何标签都可以看作是一个对象，如 body 就是一个对象。这些对象一般都有属性、事件和方法。其相关内容将在后面的项目中阐述。

（2）如果想在页面被引导后执行一个任务，可以给<body>标签添加一个 onload 事件（关于事件将在后续项目中详细介绍），此事件在 body 对象被加载完成后被触发，写法形如：

```
onload="要执行的代码";
```

（3）alert（"参数"）功能是弹出一个对话框，对话框中的内容就是该函数中设置的参数。

（4）JavaScript 对大小写是敏感的，所以 alert()必须全部小写。

任务分析

当网页被打开执行时，实际上就是触发了页面的引导事件，在该事件中使用 alert 函数执行一个弹出对话框的任务即可。

任务实现

（1）建立一个页面文件“对话框.html”（为方便测试，本书都使用静态页面）。

（2）在<body>标签内部输入以下代码：

```
<body onload="alert('你好！')">
```

（3）运行网页，可以看到如图 1.1 所示的提示对话框。

图 1.1　弹出对话框

代码解释

当网页运行时，加载了 body 对象，从而触发了 body 对象的 onload 事件，因为 onload 事件就是当 body 对象加载完毕后被触发的，其触发的处理结果是执行 alert（'你好！'），从而弹出对话框。当 body 被加载后，onload 属性就是让 body 响应 onload 事件。

onload 后面的功能代码已经被一个双引号包含，所以 alert()函数中的参数就不能再用双引号包含了，而应该用单引号。除了输出的内容部分外，所有的语法符号都要使用半角英文符号。

小贴士

（1）JavaScript 的编辑工具有很多，任何文本编辑器都可以编辑，如记事本、Notepad、EditPlus 等。当然，使用更为专业的编辑工具可以起到事半功倍的效果，如 Dreamweaver、Aptana 等。本书使用的工具为 Dreamweaver，如图 1.2 所示。

（2）使用浏览器测试 JavaScript 运行效果时，由于各种浏览器对其支持程度有别，所以，有些时候同一段 JavaScript 代码在不同的浏览器或者同一种浏览器的不同版本下运行的效果会有所不同，这一点会在后面的项目中介绍。本书所有案例均在 Windows 7 系统环境下的 IE 11.0。

浏览器中运行。

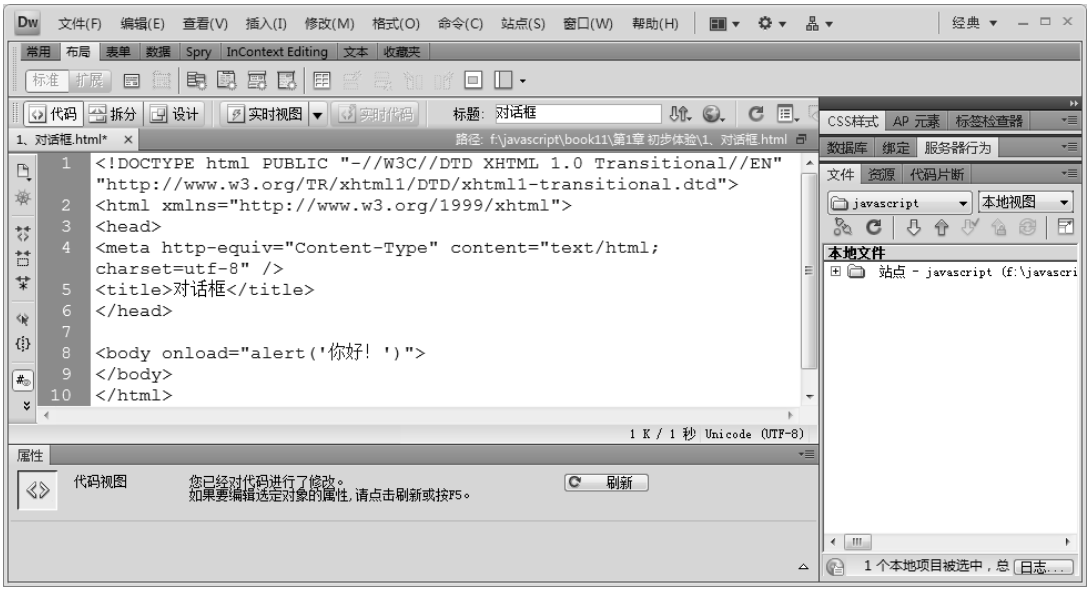

图 1.2　Dreamweaver 编辑界面

任务 2　在页面上打印欢迎词

提出任务

当页面被打开时，在网页上打印一串欢迎词。

知识预备

（1）document 表示的是文档对象，每个载入浏览器的 HTML 文档都会成为 Document 对象（关于该对象将在后续项目中详细介绍）。

（2）document 对象有很多方法，write 方法是其中之一，表示在文档中打印信息内容。

任务分析

与本项目任务 1 类似，当网页被打开执行时，触发页面的引导（onload）事件，在该事件中使用 document 对象的 write 方法执行一个打印任务即可。

任务实现

（1）建立一个页面文件“ 打印.html ”。

（2）在<body>标签内部输入以下代码：

```
<body onload="document.write('欢迎光临本站！')">
```

（3）运行网页，可以看到如图 1.3 所示信息。

图 1.3　页面打印信息

代码解释

当网页运行时，触发了 body 对象的 onload 事件，而 onload 事件的处理结果是执行 document.write（'欢迎光临本站！'），从而在网页上打印该信息。

任务 3　先弹出对话框再打印信息

提出任务

当页面被打开时，先弹出问候对话框，用户单击对话框中的“ 确定 ”按钮关闭对话框后，再在页面上打印欢迎词。

知识预备

在 JavaScript 中，如果功能块有多个语句，则这些语句中间用分号隔开。实际上，每一个完整的功能语句结束处都应该写上分号，哪怕只有一条语句，也应该写上。

任务分析

该任务综合了本项目的任务 1 和任务 2，需要注意事件中有多条语句时的写法。

任务实现

（1）建立一个页面文件“ 对话框和打印.html ”。

（2）在<body>标签内部输入以下代码：

```
<body onload="alert('你好！');document.write('欢迎光临本站');">
```

（3）运行网页，可以看到先弹出了对话框，如图 1.4 所示。单击对话框中的“ 确定 ”按钮，该对话框关闭，同时在页面上打印了欢迎信息，如图 1.5 所示。

代码解释

当网页运行时，触发了 body 对象的 onload 事件。在事件的执行任务中，先执行第一句，弹出对话框。此时，如果用户不单击对话框中的“ 确定 ”按钮，alert()函数未执行结束，那么第二句不会执行，此时，在页面上是看不到欢迎信息的。只有当用户单击了“ 确定 ”按钮后，才能看到页面上打印的欢迎信息。

图 1.4　弹出对话框

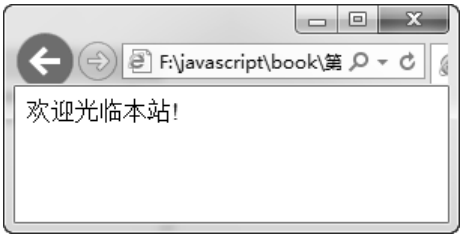

图 1.5　在页面上打印信息

小贴士

读者可以试着把执行任务中的这两条语句调换一下顺序，观察运行结果有什么不同。

任务 4　单击按钮执行任务 3

提出任务

页面上有一个按钮，单击此按钮后，先弹出问候对话框，用户单击对话框中的“ 确定 ”按钮关闭对话框后，再在页面上打印欢迎词。

知识预备

document.write()执行时，将重写当前页面。所以，页面上原有的内容将全部消失，只留下 document.write()写下的内容。在本任务中，页面中最后看不到原来的“ 问候 ”按钮就是这个原因。

任务分析

该任务需要页面与用户的交互，实际上就是由用户触发单击（onclick）事件，在该事件中执行相应的任务。

任务实现

（1）建立一个页面文件“ 单击按钮.html ”。

（2）在<body>标签对中编写按钮代码，并在按钮代码中添加 onclick 事件属性，并编写执行任务代码：

```
   <input type="button"name="button"id="button" value="问候"onclick="alert('你好！
');document.write('欢迎光临本站！');"/>
```

（3）运行网页，可以看到页面上有一个“ 问候 ”按钮，单击此按钮，先弹出“ 你好 ”对话框，单击对话框中的“ 确定 ”按钮，对话框关闭后，在页面上打印了“ 欢迎光临本站！”的信息。注意：页面上的“ 问候 ”按钮已经看不到了。

代码解释

本任务和本项目任务 3 类似，只不过是将要完成的任务交给了单击按钮，而不是加载页面。

按钮是一个对象，它的单击事件就是 onclick。当用户单击按钮时，触发了 onclick 事件，从而执行了事件中定义的任务。

任务 5　任务模块化

提出任务

仍然实现本项目任务 4 的需求，但是要求把任务执行代码模块化，以便随时调用。

知识预备

1．代码模块化

我们姑且将代码模块化简单理解为将要执行的代码“ 打包 ”为一个函数的形式，在需要的时机来调用，从而达到使代码结构清晰、复用性、可读性强的目的。

以下这些情形可能要求代码模块化。

① 任务代码较多或者较复杂。

② 同一个对象触发不同的事件执行相同或相似的任务。

③ 不同的对象触发某些事件执行相同或相似的任务。

2．JavaScript 与 HTML 混编

如果要将 JavaScript 代码写在触发事件之外，从而达到与 HTML 混编的目的，就必须将这些代码用<script></script>标签对包含起来。值得注意的是，包含在<script> </script>标签对中的 JavaScript 代码如果是一个函数，则此函数不会被执行，而是暂存在内存中，只有等待某个事件被触发从而导致该函数被调用之后，或者遇到调用的语句时才会被执行。如果不是一个函数，而是在函数之外的 JavaScript 语句，则会按顺序被执行。例如，以下代码块中：

```
<script type="text/javascript">
alert('你好！');
function hello()
{
alert('欢迎光临本站！');
};
alert('张三');
hello();
</script>
```

当页面被加载时，语句 alert（'你好！'）没有包含在任何函数中，应立即执行，此时可以看到“ 你好！”对话框的出现。紧接着加载函数 function hello(){}，但不会执行。遇到语句 alert ('张三')，也会立即执行，此时可以看到此对话框出现。最后遇到调用函数的语句 hello()，则函数 hello 才被执行，只到此时才看到“ 欢迎光临本站！” 对话框的出现。

3．注释

在编写 JavaScript 程序时，可能需要对一些代码进行注释。被注释的部分不会被执行。例如，上面的代码块中的第一行“ //…… ”注释的方法有以下两种。

① 用“ //…… ”标识。这种方法一次只能标识一行。

② 用“ /*……*/ ”标识。这种方法一次可以标识多行。例如：

```
/* 语句行 1；
语句行 2；
……
语句行 n；
*/
```

任务分析

要实现任务模块化，可以将任务代码放入一个自定义的函数中，然后在需要的地方调用即可。

任务实现

（1）建立一个页面文件“ 模块化.html ”。

（2）在<body>标签对中编写按钮代码如下：

```
<input type="button"name="button"id="button"value="问候"onclick="hello()"/>
```

（3）在页面的<head></head>标签内部编写代码如下：

```
<script type="text/javascript">
//下面定义了一个函数 hello
function hello()
{
    alert('你好！');
    document.write('欢迎光临本站!');
}
</script>
```

（4）运行网页，单击按钮，效果和本项目任务 4 相同。

代码解释

在此任务中，可以看到，我们并没有将详细的任务代码像本项目任务 4 一样直接写在触发的事件 onclick 的后面，而是将这些任务代码独立出来，放入一个函数模块 function hello() 中。在触发的事件 onclick 中只要调用此函数模块即可。有关函数的具体内容将在后面的章节阐述。

小贴士

<script type = "text/javascript"> 也可以写作 <script language = " javascript">，但 language 这个属性在 W3C 的 HTML 标准中已不再推荐使用，所以应尽量用任务中的写法。由此对标签包括起来的 JavaScript 代码可以出现在页面文档的任何位置。但是一般是写在<head>标签对中的。这样，当网页被运行时，此代码块被优先加载，从而提高了运行效率。

任务 6　调用灵活化

提出任务

仍然完成本项目任务 5 的需求，但是要求增加代码的灵活性。例如，改变对话框和页面打印的信息，但是不要修改函数模块代码。

知识预备

程序设计中常常通过在编写函数模块时定义参数（一般是用逗号隔开的变量列表），而在调用函数时传入具体数值的方法实现函数功能的灵活性和可扩展性。有关变量的概念以及函数参数的具体内容将在后面阐述。

任务分析

在本项目任务 4 中，我们将代码“打包”为一个函数，实现了模块化，但是还不够灵活。例如，要改变显示的内容，就必须修改函数体中的代码。如果再增加一个按钮，单击这个按钮也需要调用函数 hello()，但是要显示的内容又有不同，因此无法实现，只能另外编写一个函数供其调用。这样的程序缺乏了可扩展性，代码的复用性也比较差。可以考虑在编写函数时，设置两个负责从外界（调用者）接收数据的参数，外界（调用者）只要将具体的数据传递给函数的参数，该函数就能对参数进行相应的处理，最终返回结果。

任务实现

（1）建立一个页面文件“参数传递.html”。

（2）在<body>标签对中编写按钮代码如下：

```
   <input type="button"name="button"id="button"value="问候"onclick="hello('你好',
'欢迎光临本站！');"/>
```

（3）在页面的<head></head>标签内部编写代码如下：

```
01  <script type="text/javascript">
02  function hello(str1, str2)
03  {
04      alert(str1);
05      document.write(str2);
06  }
07  </script>
```

（4）运行网页，单击按钮，效果和本项目任务 5 相同。

代码解释

hello()函数中提供了两个参数，分别为 str1 和 str2，等待调用者传入具体的值。函数内部将对这两个参数进行相应的处理。hello（'你好'，'欢迎光临本站！'）是将具体的值字符串"你好"传给了参数 str1，将字符串"欢迎光临本站！"传给了 str2。如果要改变显示的内容，只要改变调用者传入的具体值即可，而不需要改动函数 hello()的代码。

任务 7　“项目 1”的实现

知识预备

1．JS 文件

扩展名为.js 的文件是用 JavaScript 编写的客户端脚本文件，它不是一般的网页文件，一般不能直接运行打开，而是配合网页来使用。它常常用来实现某些功能，这些功能代码可以被多个不同的网页调用。在 JS 文件中编写 JavaScript 脚本时，不要用<script>标签对包括。

2．引用 JS 文件

在页面中，只有引入 JS 文件才能使用该文件中的 JavaScript 代码。方法是在<script>标签中添加 src 属性，其值就是该 JS 文件的路径。

项目分析

很显然，当完成了本项目任务 6 后，要想完成项目就非常简单了，只要制作两个页面，分别在这两个页面中用本项目任务 6 中的方法就可以实现。但是，在项目最终实现时，希望项目中的代码复用性和可维护性更强。试想，既然这两个页面调用的函数都是一样的，那么可不可以把这个函数独立出来供大家共享呢？答案是肯定的。

项目实现

（1）建立一个文件 hello.js（注意，扩展名为.js）。

（2）在 hello.js 中编写如下函数代码并保存文件。

```
01  function hello(str1, str2)
02  {
03      alert(str1);
04      document.write(str2);
05  }
```

（3）建立两个页面，分别为 hello1.html 和 hello2.html（为简单起见，将这 3 个文件放置在同一个目录下）。

（4）在 hello1.html 中，<body>标签对中内部编写按钮代码如下：

```
   <input type="button" name="button" id="button" value="问候" onclick="hello('
张三向您问好','张三欢迎您光临本站！');" />
```

（5）在页面的<head></head>标签内部编写代码如下：

```
01  <script type="text/javascript" src="hello.js">
02  </script>
```

（6）在 hello2.html 中做同样的操作，只是将“ 张三 ”改为“ 李四 ”

（7）分别运行网页 hello1.html 和 hello2.html 即可看到项目实现的结果。

代码解释

我们把提供共享的函数 hello()的代码单独写在一个扩展名为.js 的文件中，在调用页面

hello1.html 和 hello2.html 中，在标签<script>内部添加 src 属性，把 hello.js 文件引入。这样在调用页面就可以调用 hello.js 内部的函数了。

小贴士

hello.js 文件可以放置在站点目录的任何位置，在调用页面用 src 引入时一定要写清楚路径，而且最好用相对路径。

拓展实训

在网页中插入一张图片，完成以下需求：

（1）当用鼠标单击图片时，弹出对话框“你好！”，单击对话框中的“确定”按钮后，在页面上打印“欢迎光临！”，如图 1.6 和图 1.7 所示。

（2）要求用三种编写方式来实现。

① 把代码直接写在<img>标签内。

② 在同一个页面内部采用函数调用的方式。

③ 把函数写在 JS 文件中，然后调用。

图 1.6　单击图片弹出对话框

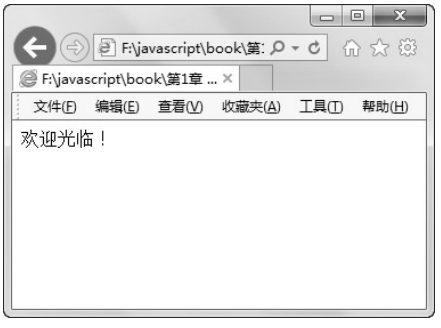

图 1.7　在页面上打印信息