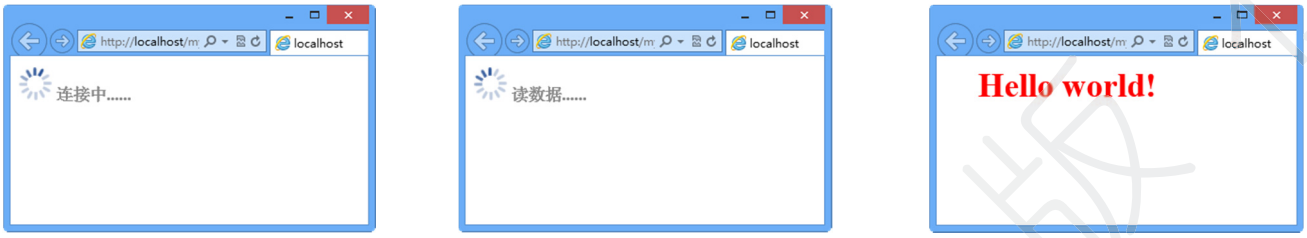
21.3 扫码实战

本节将结合几个典型案例从不同侧面介绍 Ajax 的应用。为便于学习，在设计实例时，侧重实际需要，提炼核心技术，避免重复。每个示例都涉及 XMLHttpRequest 创建操作，由于该函数在上一节中已详细说明，本节就不再提及。

21.3.1 Ajax 交互提示

使用 Ajax，浏览器不再刷新页面，这样用户就无法知道页面的交互进程，网页是否与后台联系，数据是否加载更新？当打开的一个空白页，也许数据还没有被加载完毕，但是如果没有提示信息，会让用户无所适从，甚至产生误解。为了避免此类问题，让用户知道当前操作的进程，应该提供实时提示信息，本示例演示效果如图 E21.1 所示（index.html）。



连接到后台　　读取数据　　响应成功

图 E21.1　Ajax 异步交互过程

设计思路：

这种提示信息设计其实很简单，用户只需要根据 readyState 属性即可准确进行判断：

- 当值为 0 时，表示还没有使用 open()方法打开连接；
- 当值为 1 时，表示还没有使用 send()方法发送请求，也就是说还没有连接上后台；
- 当值为 2 时，表示已经使用 send()方法发送请求，但是还没有接到响应信息；
- 当值为 3 时，表示已经开始接收响应数据，但是还没有接收完毕；
- 当值为 4 时，表示数据接收完毕，整个请求响应的交互过程成功。

下面就借助 readyState 属性来制作一个 Ajax 交互过程的提示信息示例，示例效果如图 21.18 所示。其中“Hello world!”字符串是客户端请求服务器时发送过去的，然后服务器接受请求，并把该字符串响应给客户端，从而在浏览器中显示“Hello world!”字符串信息。

【操作步骤】

第 1 步，启动 Dreamweaver，新建文档，保存为 index.html。在文档中构建一个简单的一级标题标签，用来承载服务器响应的信息。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<style type="text/css">
h1 {              /*设计标题字体颜色和大小*/
    color:#888;
    font-size:16px;
}
h1 span {         /*定义标题内 span 元素字体大小和颜色，增加补白*/
    padding:1em;
    color:red;
    font-size:32px;
}
</style>
</head>
```

```
<body>
<h1 id="info"></h1>
</body>
</html>
```

第 2 步，设计思路：把“Hello world!”字符串发送到服务器端，然后服务器接收该字符串信息，再把该字符串信息响应给客户端，最后由 JavaScript 接收并插入到页面中。因此，需要设计一个简单的后台接收和处理文件（test.asp）。

```
<%@LANGUAGE="VBSCRIPT" CODEPAGE="936"%>
<%
dim data                '定义变量
data = Request.QueryString("data")
'接收从客户端传递过来的字符信息，并存储在 data 变量中
'定义响应信息页的字符编码，由于信息中包含中文字符，所以这里设置为简体中文。默认为 UTF-8，如果不设置中文，可能会出现乱码
Response.AddHeader "Content-Type","text/html;charset=gb2312"
Response.Write data
'把字符信息写回（响应给）客户端浏览器
%>
```

第 3 步，在页面初始化（页面加载）完毕时使用 XMLHttpRequest 对象的 open()方法打开与服务器的连接，然后使用 send()方法发送请求。在请求 URL 之后以查询字符串的形式附带传递一个字符串信息。同时在 onreadystatechange 中绑定回调函数 updatePage，以便使用该函数处理服务器响应的数据。

```
<script>
window.onload = function(){
//定义页面初始化处理函数
    var str = "test1.asp?data=Hello world!";
//设置请求的 URL 以及附加的查询字符串信息
    request.open( "GET", str, true );                    //设置打开异步请求连接
    request.onreadystatechange = updatePage;      //绑定回调函数
    request.send( null );                                    //发送异步请求
}
</script>
```

第 4 步，定义回调函数。在该函数体内利用 readyState 属性实时跟踪异步交互的过程（注意大小写）。

```
function updatePage(){                                   //回调函数
    var info = document.getElementById( "info" );
//获取页面中的标题标签
    if( request.readyState == 1 ){
//当前状态，还没有连接上
        info.innerHTML = "<img src='images/loading.gif’  />  连接中.....";
//显示提示信息
    }
    else if( request.readyState == 2 || request.readystate == 3 ){
//当前状态，正在响应数据
        info.innerHTML = "<img src='images/loading.gif’  />  读数据.....";
//显示提示信息
    }
    else if( request.readyState == 4 ){
//当前状态，交互胜利完成
        if( request.status == 200 ){
//请求中 HTTP 状态码
            info.innerHTML = "<span>" + request.responseText + "</span>";
//显示提示信息
        }
        else
```

```
        alert( request.status );
    //如果交互失败，则弹出 HTTP 状态码
        }
    }
    </script>
```