第2章补充内容4

AJAX

AJAX

Asynchronous JavaScript And XML

(异步的 JavaScript 和 XML)

异步交互式网页开发技术

目录

4.1 为什么使用Ajax

4.1.1 典型的WEB应用系统工作原理

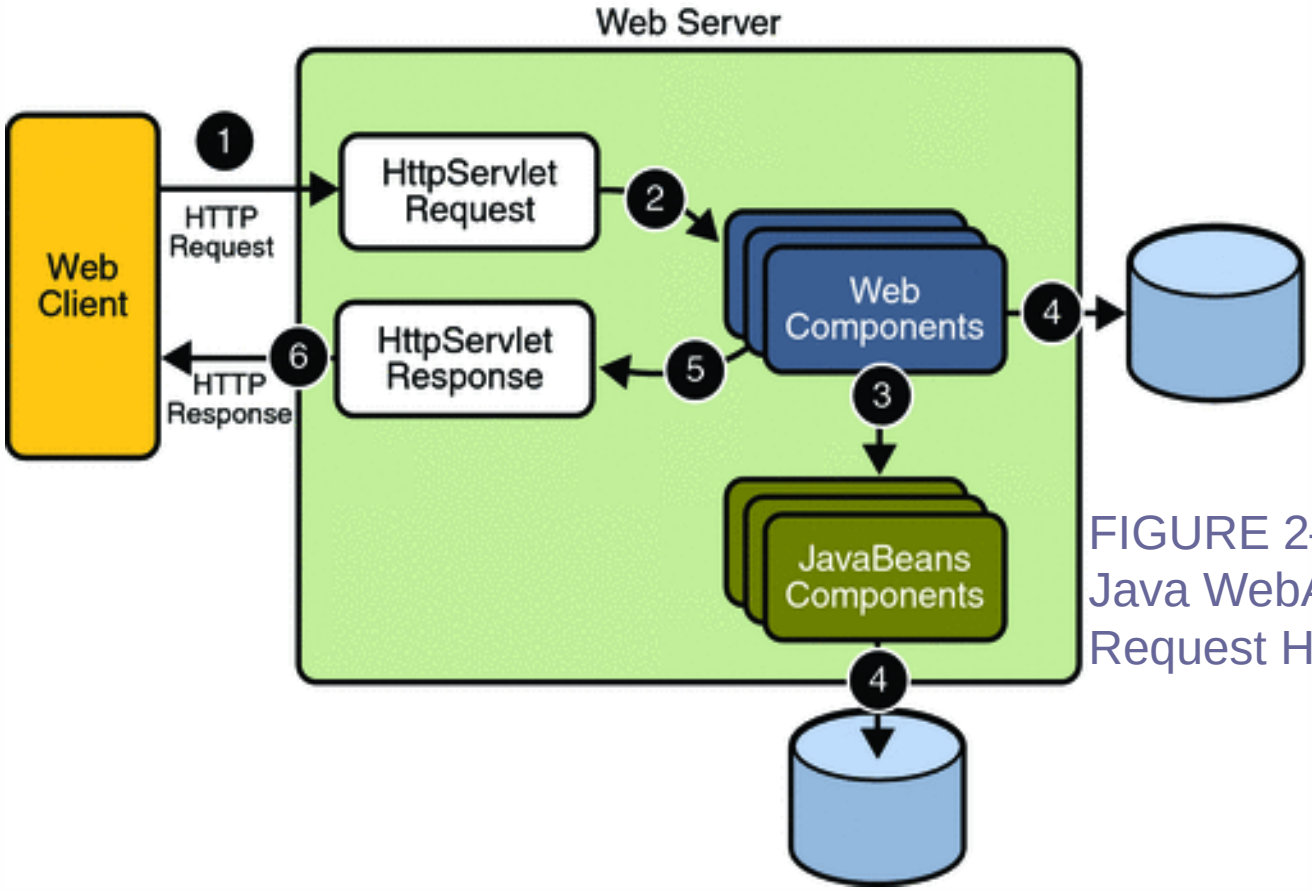


FIGURE 2–2
Java WebApplication Request Handling

4.1 为什么使用Ajax

状态码：成功

例如：

HTTP/1.1 200 ok ← 状态行

Last-Modified:Mon,April 16,2021 2:03:37 AM
Date:Mon,Mon, April 23,2007 12:01:42 AM
Status:200
Content-Type:text/html
Content-Length:186

← 一个空行

响应报头

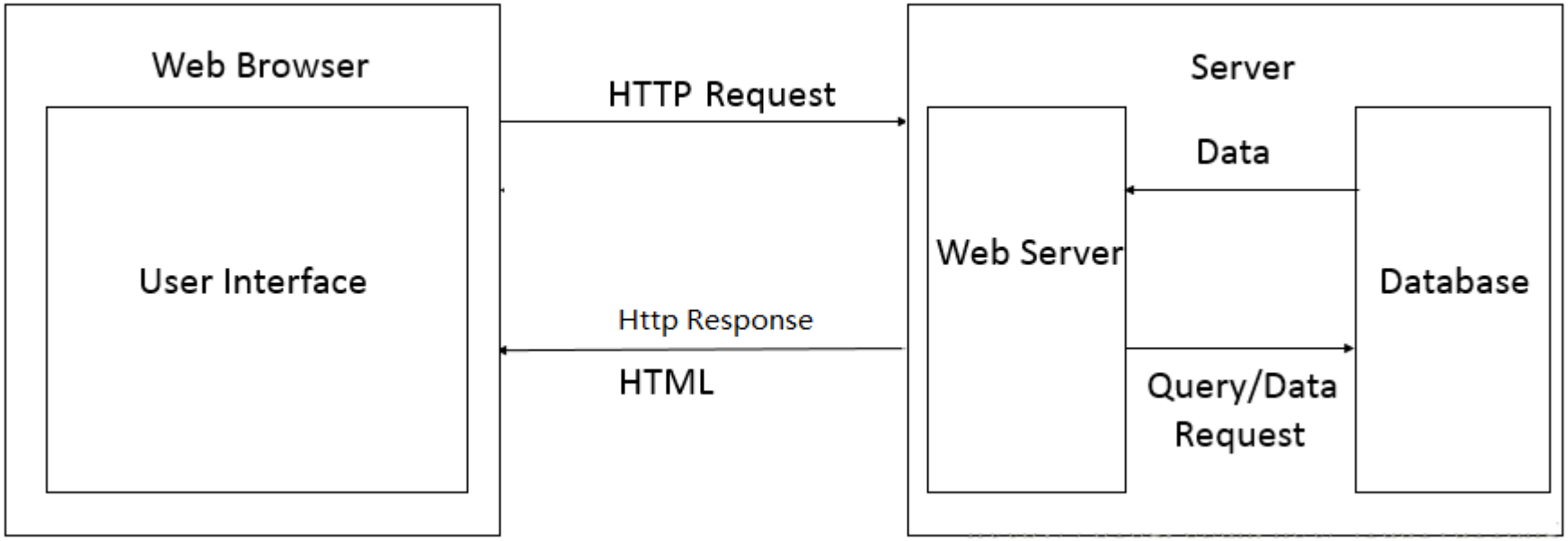
```
<html>
<head>
<meta charset="utf-8">
<title>  </title>
</head>
<body>
<p>
JavaScript 能够直接写入 HTML 输出流中:
</p>
<script>
document.write("<h1>这是一个标题</h1>");
document.write("<p>这是一个段落。</p>");
</script>
</body>
</html>
```

响应主体

4.1 为什么使用Ajax

4.1.1 典型的WEB应用系统工作原理

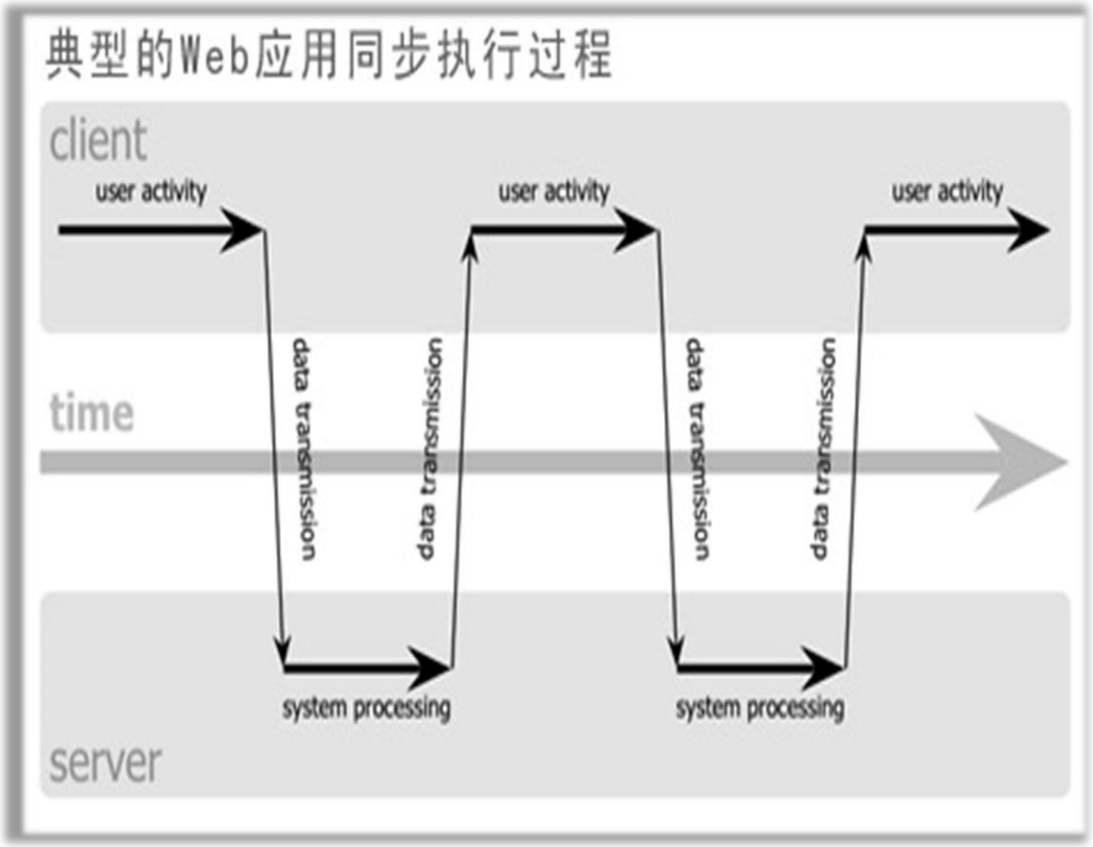
- 典型的WEB应用系统模型



4.1 为什么使用Ajax

4.1.1 典型的WEB应用系统工作原理

- 传统的Web应用采用同步交互过程。



同步：发送请求后等返回结果

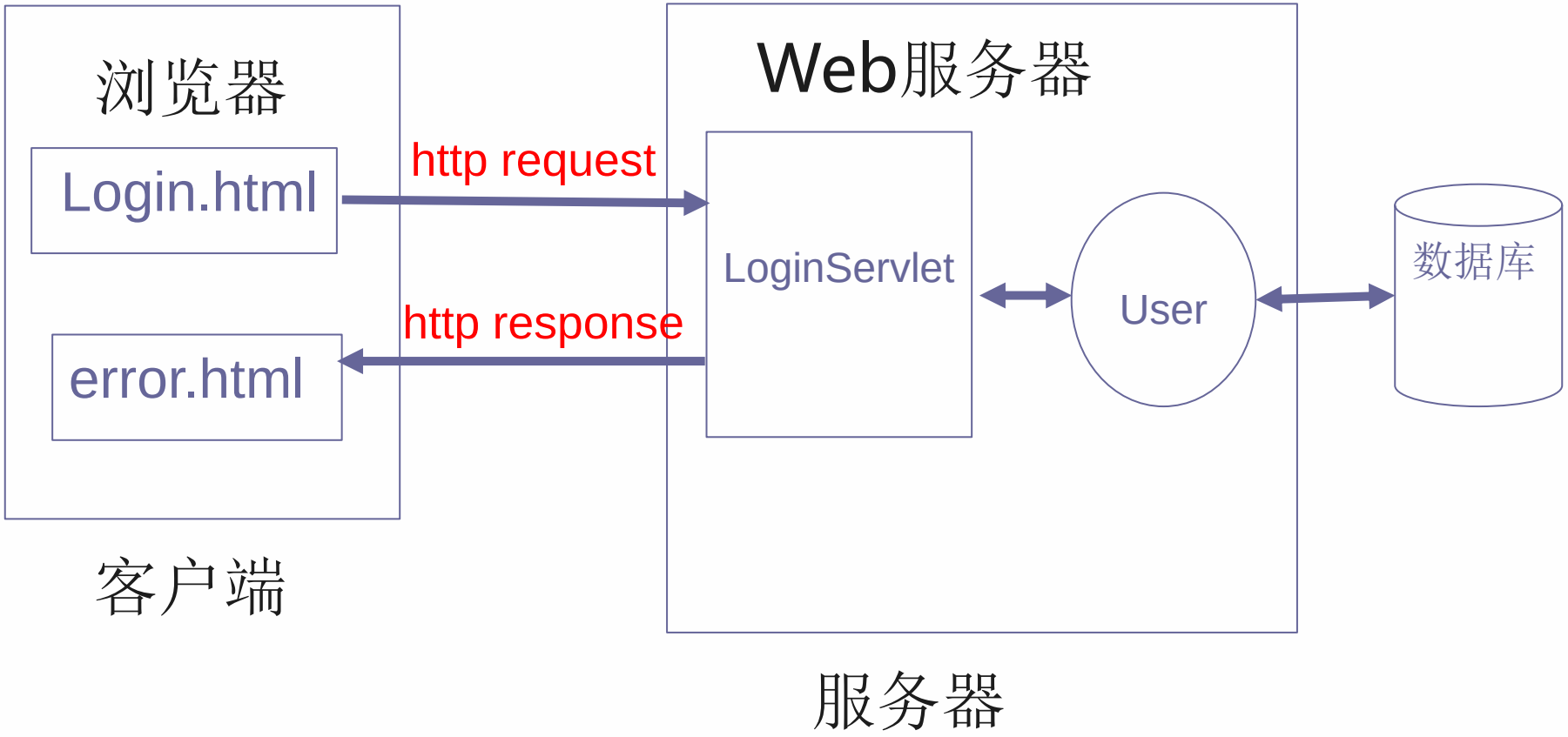
4.1 为什么使用Ajax

4.1.1 典型的WEB应用系统工作原理

- 传统的Web应用采用同步交互过程。
  - 传统的Web应用中，用户单击一个链接后，待需要等待服务器端返回一个新的页面刷新显示 。
  - 如果仅仅需要改变页面的某一部分的内容，也不得不刷新整个页面。

4.1 为什么使用Ajax

4.1.2 例：系统登录用例



4.1 为什么使用Ajax

4.1.2 例：系统登录用例

- Login.html

4.1 为什么使用Ajax

4.1.2 例：系统登录用例

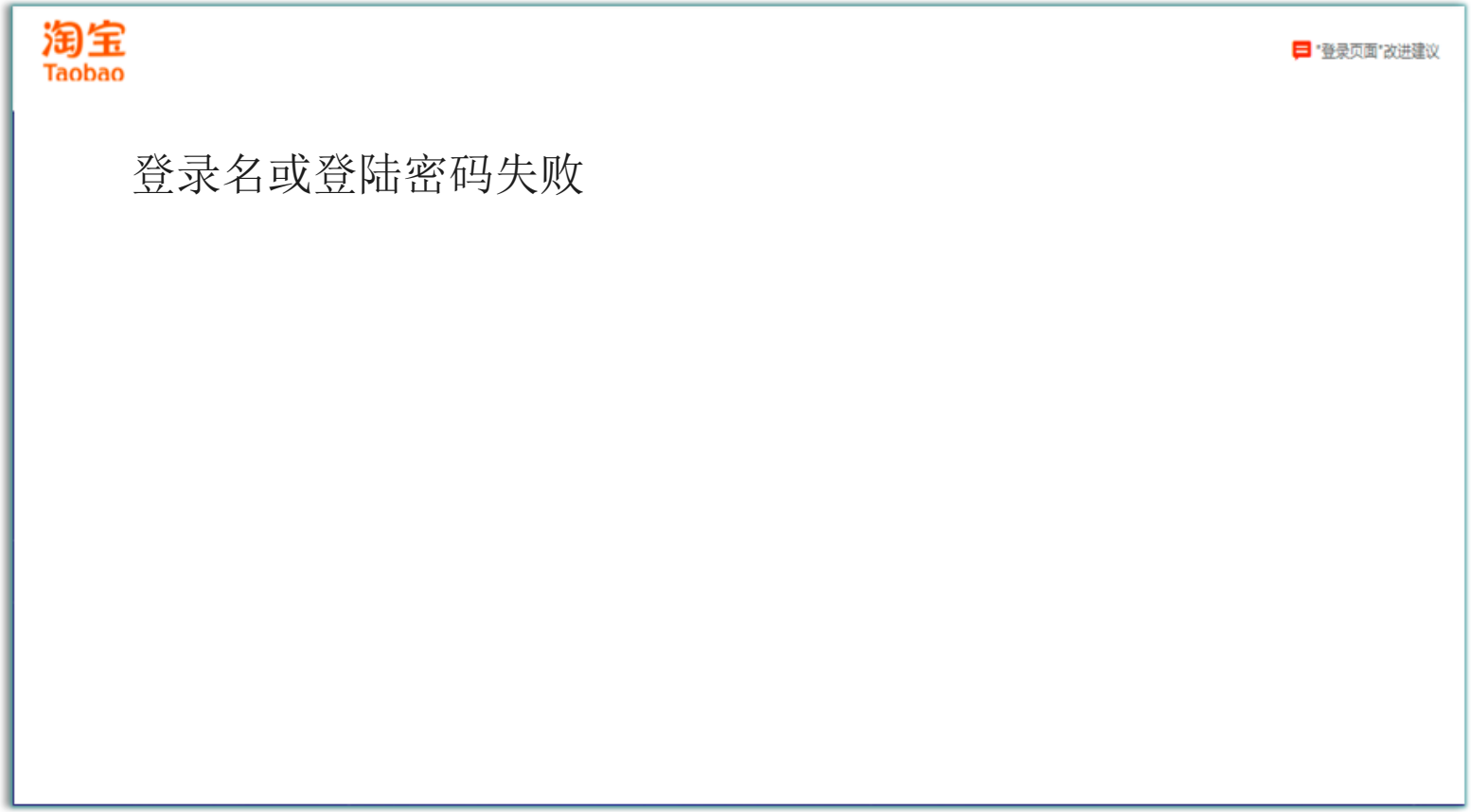
- 登录页面输入错误用户名和密码

4.1 为什么使用Ajax

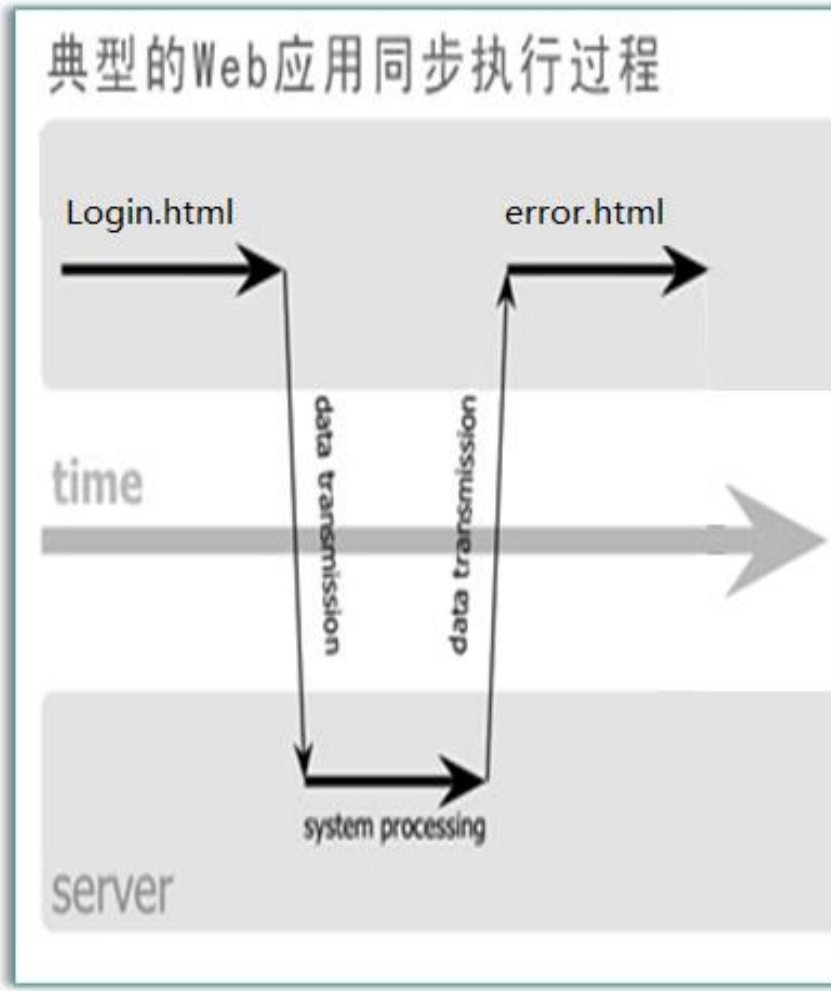
4.1.2 例：系统登录用例

- 刷新整个页面：重新显示一个页面
- error.html

登录失败，刷新整个页面。



4.1.2 例：系统登录用例


典型的Web应用同步执行过程
Login.html
error.html
data transmission
data transmission
time
system processing
server

4.1 为什么使用Ajax

4.1.3 例：系统登录用例（使用Ajax）

- 登录页面输入错误用户名和密码

4.1 为什么使用Ajax

4.1.3 例：系统登录用例（使用Ajax）

- 无刷新：不刷新整个页面，只刷新局部。

4.1 为什么使用Ajax

登录成功，没有用Ajax技术，刷新整个页面。

4.2 Ajax原理

4.2.1 Ajax：定义

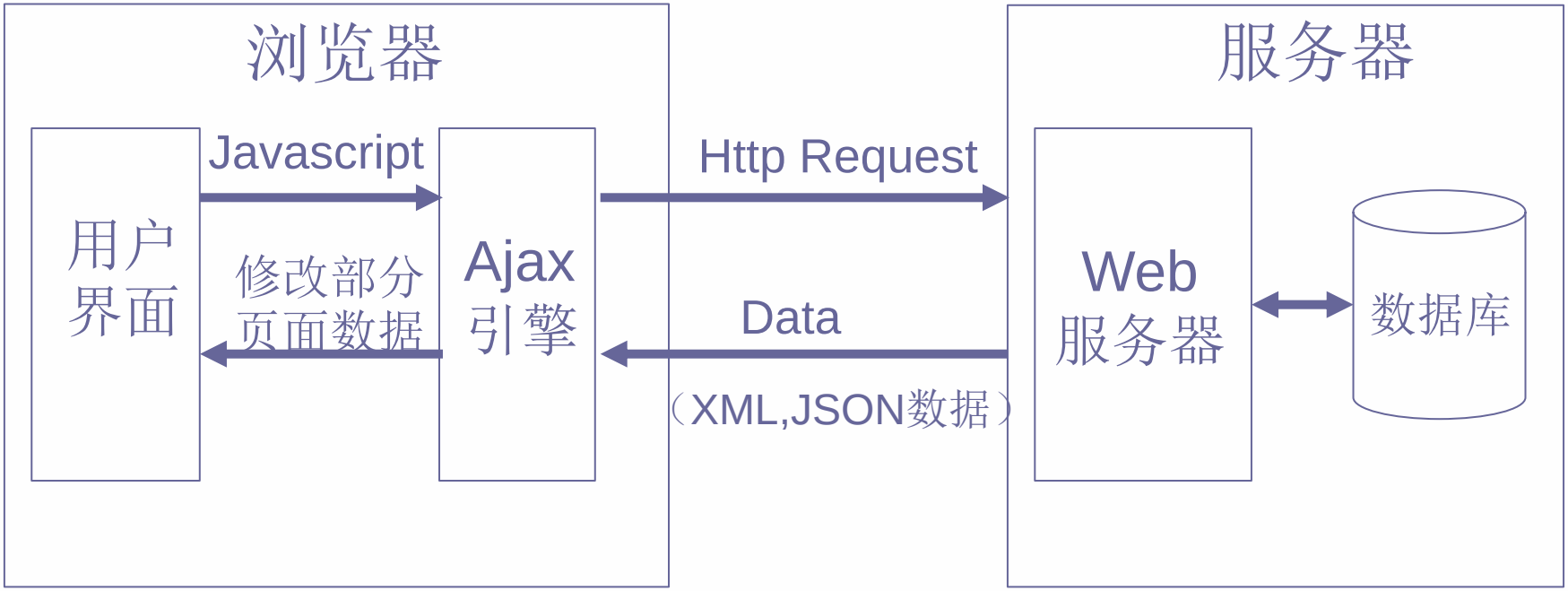
- Ajax： Asynchronous JavaScript And XML（异步的 JavaScript 和 XML）

Asynchronous 异步的	JavaScript	And	XML

- 异步：发送请求后不等返回结果，由回调函数处理结果
- JavaScript：向服务器发起请求，获得返回结果，更新页面
- XML：封装数据

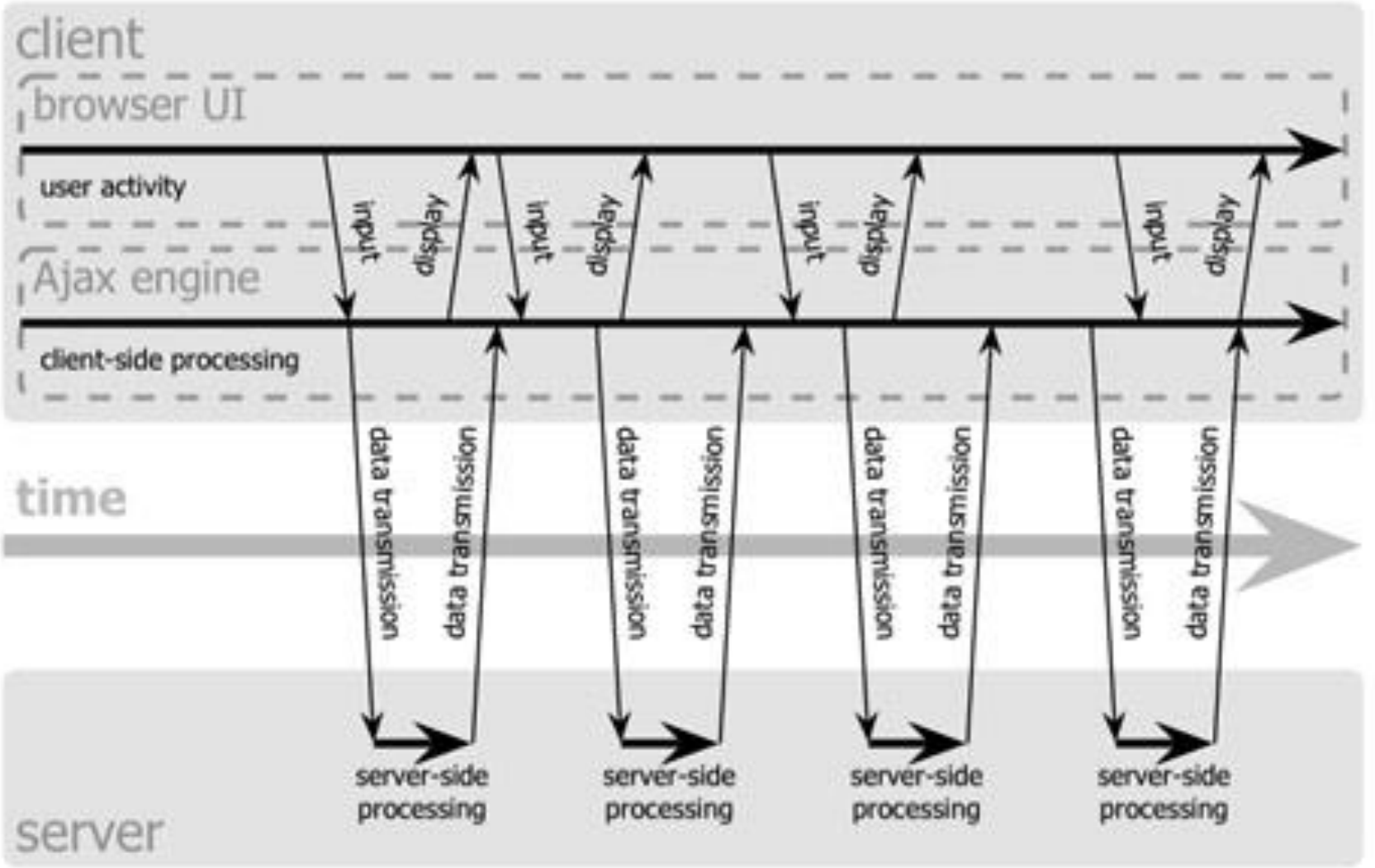
4.2 Ajax原理

4.2.2 Ajax异步交互模型



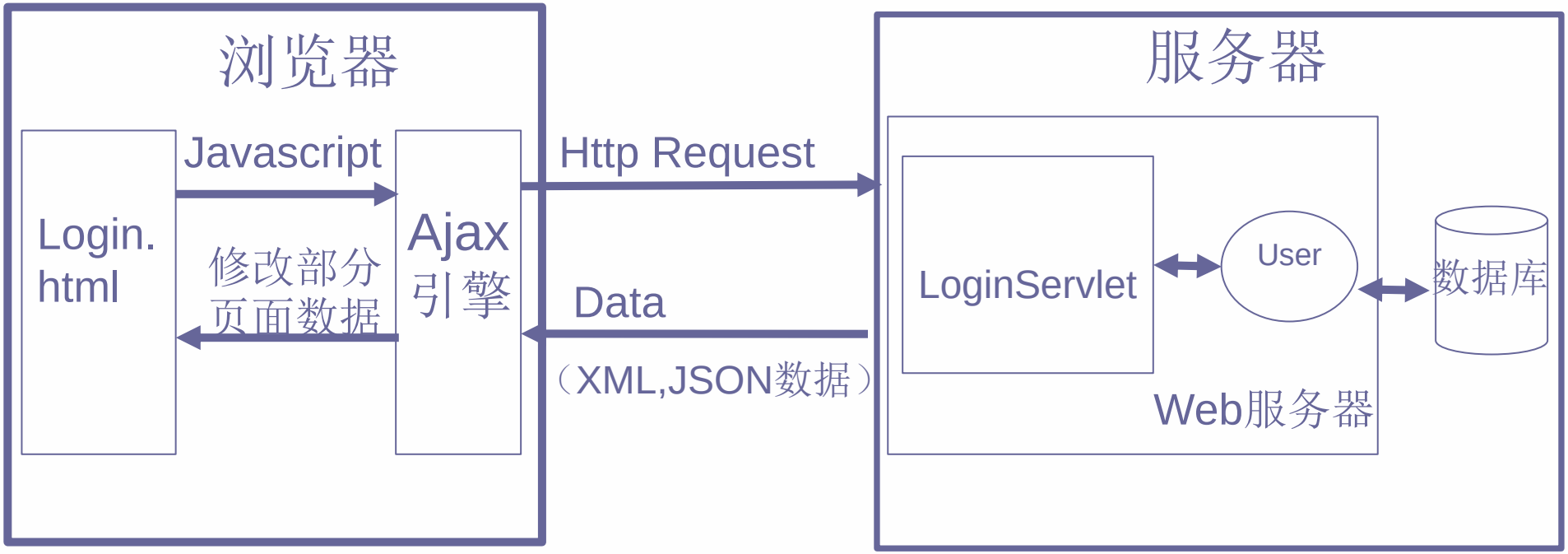
4.2 Ajax原理

4.2.3 基于AJAX的异步交互过程



4.2 Ajax原理

4.2.4 例：系统登录用例



4.3 Ajax技术

4.3.1 Ajax技术

- AJAX（Asynchronous JavaScript and XML）它是一种由多种技术组合的技术。

  包括Javascript、DOM、XML、XMLHttpRequest 等

  - XMLHttpRequest对象用于进行异步数据读取
  - JavaScript/DOM 实现动态显示和交互
  - XML作为转换数据的格式

4.2 Ajax原理

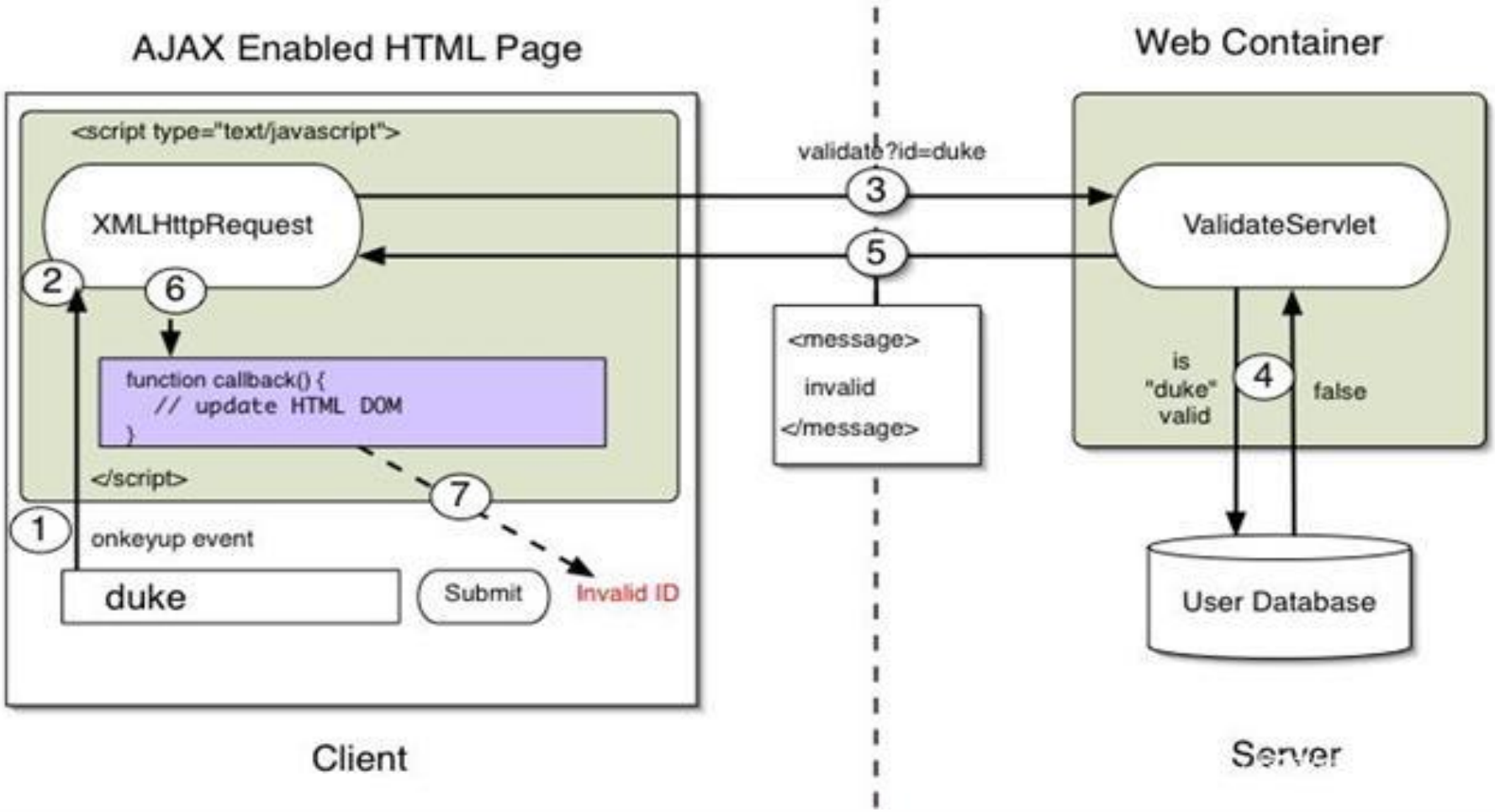
4.3.1 Ajax技术

- 说明
  - AJAX不是一种新的编程语言，而是一种用于创建交互性更强的 Web 应用程序的技术。
  - Ajax的核心是JavaScript对象XmlHttpRequest。
    - 该对象在Internet Explorer 5中首次引入，它是一种支持异步请求的技术。
    - 简而言之，XmlHttpRequest使您可以使用JavaScript向服务器提出请求并处理响应，而不阻塞用户。

4.3 Ajax技术

4.3.1 Ajax技术

- 页面请求执行过程(采用Ajax技术)



ajax1.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script>
    function loadXMLDoc()
    {
            var xmlhttp;
            if (window.XMLHttpRequest)
            {
              xmlhttp=new XMLHttpRequest(); //  IE7+, Firefox, Chrome 浏览器执行代码
            }
            else
            {
              xmlhttp=new ActiveXObject("Microsoft.XMLHTTP"); // IE6, IE5 浏览器执行代码
            }
            xmlhttp.onreadystatechange=function()
            {
               if (xmlhttp.readyState==4 && xmlhttp.status==200)    {
                   document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
                }
            }
            xmlhttp.open("GET","/try/ajax/ajax_info.txt",true);
            xmlhttp.send();
    }
</script>
</head>
<body>

<div id="myDiv"><h2>使用 AJAX 修改该文本内容</h2></div>
<button type="button" onclick="loadXMLDoc()">修改内容</button>

</body>
</html>
```

4.3 Ajax技术

4.3.2 XMLHttpRequest

- AJAX核心对象是XMLHttpRequest，该对象在JavaScript中可用于构建异步的后台服务的调用。
- 通过这个对象，AJAX可以像桌面应用程序一样只同服务器进行数据层面的交换，而不用每次都刷新界面。
- 不同的浏览器它的构建方式并不一样：
  - Mozilla,NetScape：

```
var httpRequest = new XMLHttpRequest();
```

  - IE：

```
var httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
```

4.3 Ajax技术

4.3.2 XMLHttpRequest

- XMLHttpRequest的使用

使用XMLHttpRequest异步调用后台服务的基本步骤:

- 初始化XMLHttpRequest对象
- 指定响应处理函数(回调方法)
- 发出HTTP请求
- 处理服务器返回的信息

4.3 Ajax技术

4.3.2 XMLHttpRequest

- 初始化XMLHttpRequest

不同的浏览器构建方式并不一样，示例展示了适合IE和Mozilla浏览器的构建方式：

```
if (window.XMLHttpRequest) {
        // Mozilla, Safari, ...
        http_request = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE
        http_request = new ActiveXObject("Microsoft.XMLHTTP");
}
```

4.3 Ajax技术

4.3.2 XMLHttpRequest

- 指定响应处理函数(回调方法)

回调方法在服务器端返回信息给客户端时被调用，只需将回调方法指定给XMLHttpRequest对象的onreadystatechange属性即可，示例中展示了两种设置方法。

```
var processRequest = function (){ … }
http_request.onreadystatechange = processRequest;
```

```
http_request.onreadystatechange = function(){
    …
}
```

4.3 Ajax技术

4.3.2 XMLHttpRequest

- 发出HTTP请求

在发送请求前需要调用XMLHttpRequest的open方法打开链接，之后可通过其send方法发送请求。

```
http_request.open('GET', 'htt://localhost:8080/useprj/findProduct?id=01', true);
```

open方法的三个参数分别为：

- 请求的方式GET,POST,HEAD
- 请求的路径
- 是否异步请求，如果指定为异步，在请求发送后，浏览器继续执行，否则等待。

4.3 Ajax技术

4.3.2 XMLHttpRequest

- 发出HTTP请求
  - XMLHttpRequest的send方法用于向服务器发送请求。

```
http_request.send(…);
```

  - 请求如果是以POST方式发出，send方法的参数对应发送到服务器的数据，如果数据为上传的文件，需要设置请求的头信息，例如：

```
http_request.setRequestHeader(“Content-Type”,”
multipart/form-data”);
```

4.3 Ajax技术

4.3.2 XMLHttpRequest

- 处理服务器返回
  - 请求发送后，浏览器会根据请求或响应的状态调用XMLHttpRequest的**回调方法**，状态信息保存在XMLHttpRequest对象的readyState属性中。
  - 不同的readyState值对应响应的不同阶段，当readyState的值为4时表示服务器响应完成。

示例代码中根据返回的状态及响应的结果状态，执行处理：

```
if (http_request.readyState == 4) {  // 信息已经返回，可以开始处理
    if (http_request.status == 200) {
        // 页面正常，可以开始处理信息
    } else {  // 页面有问题}
} else {    // 信息还没有返回，等待}
```

4.3 Ajax技术

4.3.2 XMLHttpRequest

- 处理服务器返回

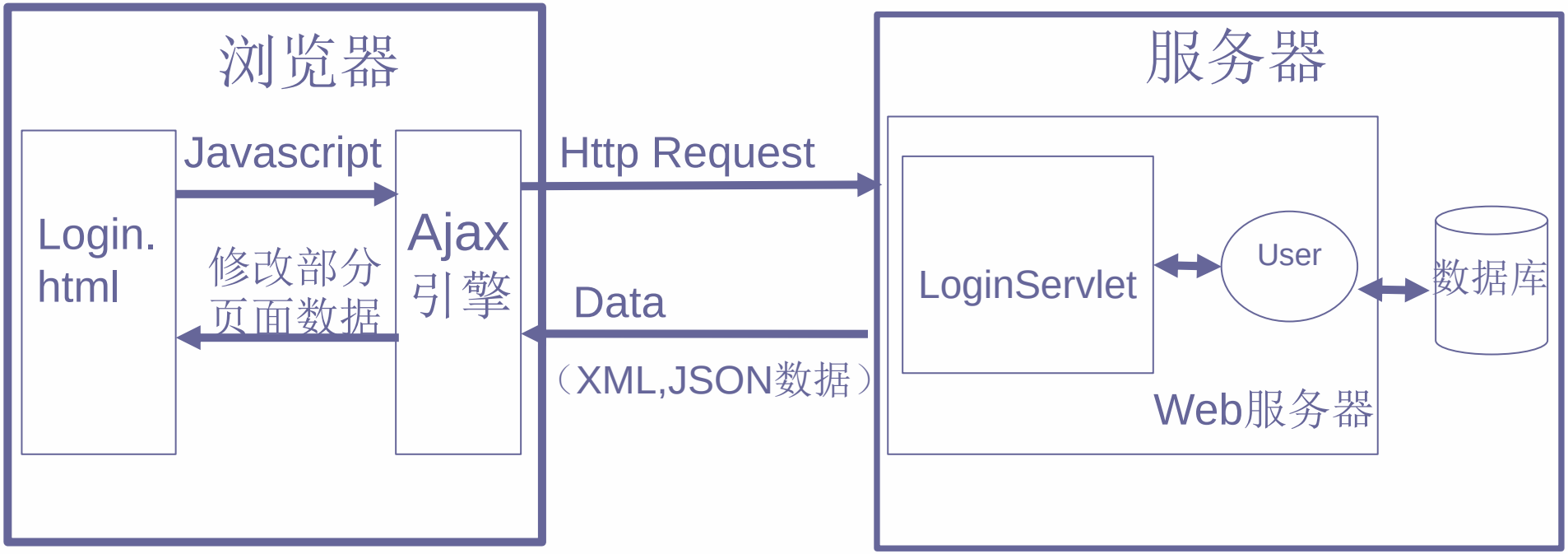
XMLHttpRequest成功返回的信息有两种处理方式：

- responseText该属性以**字符串的形式**返回响应的值
- responseXML该属性将返回结果作为一个XML的DOM文档返回，可以执行DOM处理。

示例中获取响应的文本并显示：

```
if (http_request.readyState == 4) {
    if (http_request.status == 200) {
        alert(http_request.responseText);
    }
}
```

4.4 例子：用户登录（采用Ajax技术）



4.4 例子： 用户登录（采用Ajax技术)

示例展示了无刷新页面的验证用户代码的客户端HTML代码:

```
<h2>用户登录</h2>
<div id="msg"></div>
<form name="loginform">
    帐号：<input name="userid" type="text"/>
    密码：<input name="pwd" type="password
    <input value="登录" type="button" onclick="checkUserLogin()"/>
 </form>
```

4.4 例子： 用户登录（采用Ajax技术）

验证用户代码的客户端AJAX代码： ajaxjavascript.html

```
function checkUserLogin(){
 var request = null;
 try{   request = new ActiveXObject("MSXML.XMLHTTP");
 }catch(e){  request = new XMLHttpRequest();  }

 request.onreadystatechange = function(){
        if(request.readyState == 4){  if(request.status == 200){
        if(request.responseText == 'true'){
               document.write("<h1>欢迎光临!</h1>");
        }else{
              document.getElementById("msg").innerHTML= request.responseText;
              document.loginform.reset();
        } } } };

       request.open("post", "LoginServlet", true);
 var userid = document.loginform.userid.value;
 var pwd = document.loginform.pwd.value;
 request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
 request.send("userid=" + userid + "&pwd=" + pwd);
 }
```

4.4 例子： 用户登录（采用Ajax技术)

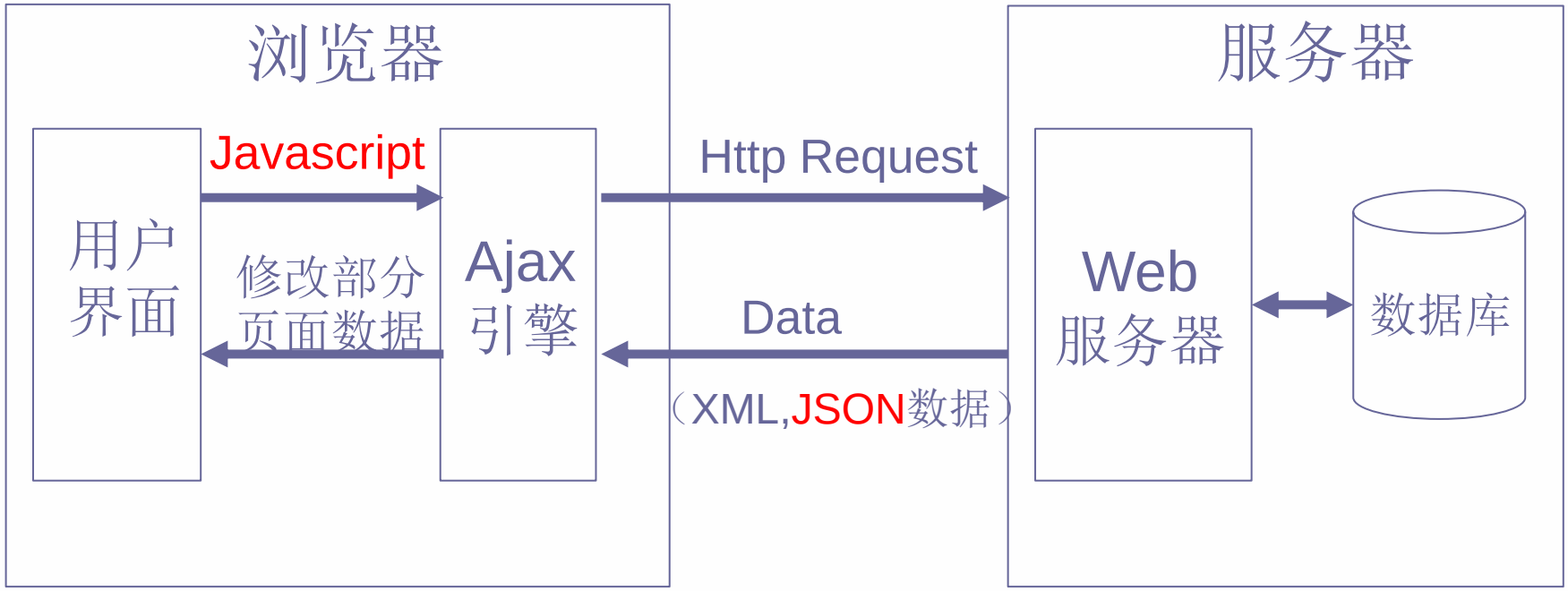
```
request.setCharacterEncoding("utf-8");
response.setContentType("text/html; charset=utf-8");
PrintWriter out = response.getWriter();
String userid = request.getParameter("userid");
String pwd = request.getParameter("pwd");
if("jerry".equals(userid) && "123".equals(pwd)){
    out.write("true");
}else{
    out.write("登录失败，用户名或密码不对！ ");
}
```

4.5 AJAX异步处理的优点

- 减轻服务器的负担，AJAX一般只从服务器获取只需要的数据。
- 无刷新页面更新，减少用户等待时间。
- 更好的客户体验，可以将一些服务器的工作转移到客户端完成，节约网络资源，提高用户体验。
- 基于标准化的对象，不需要安装特定的插件绝大多数的游览器都能执行
- 彻底将页面与数据分离。

进一步学习...

Ajax异步交互模型



jQuery 极大地简化了 JavaScript 编程。

JSON（JavaScript Object Notation JavaScript 对象标记法）：
轻量级的数据交换格式。

https://www.w3school.com.cn/

thanks