Session 19

Ajax

1

Reading & Reference

- Reading
  - Sun Tutorial

    java.sun.com/developer/technicalArticles/J2EE/AJAX/
- Reference
  - XMLHttpRequest object

    en.wikipedia.org/wiki/Xmlhttprequest

    www.w3.org/TR/XMLHttpRequest/
- JavaScript by David Flanagan, O'Reilly Press (Available on-line through Safari textbooks), Chapter 20.

2

Learning Goals

- Understand the architecture of Ajax
- Understand the XMLHttpRequest object
- Understand how to develop an Ajax application

In the next session, you will learn how jQuery wraps the XMLHttpRequest object to make it easier to use

3

What is Ajax?

- Asynchronous JavaScript Technology and XML
- Allows incremental update of Web pages within the browser
- Not dependent on any given language or data exchange format, but works well with
  - Xhtml
  - JavaScript

4

Examples

- Microcontent
  - http://tiddlywiki.com/
  - http://phrogz.net/JS/Tabtastic/index.html
- Image update
  - http://www.couloir.org/
  - http://www.tartanmaker.com

5

Ajax Uses

- Real-time form data verification
- Autocompletion
- Filtering & sorting (e.g., sorted table columns)
- Master details (deep tree navigation)
- Expanded user interface controls (e.g., voting)
- Refreshing data on the page (e.g., news/sports)
- Rapid user-to-user communications

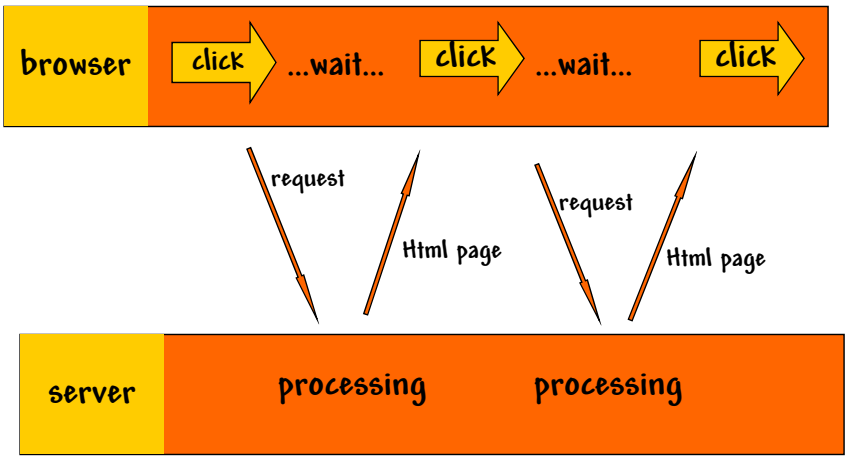
6

Ajax Limitations

- Complexity (development and debugging)
- Non-standard XMLHttpRequest object
- Viewable source code
- Download of sizeable JavaScript libraries

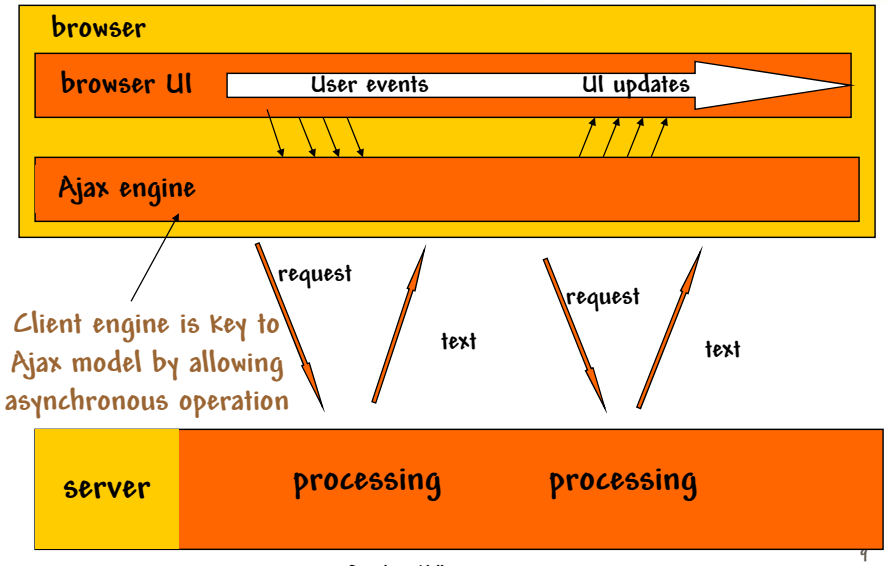
7

Classic Browser/Server Interaction



8

Ajax Browser/Server Interaction



9

Typical Ajax Interaction

1. Client event occurs
2. XMLHttpRequest object is created
3. XMLHttpRequest object calls the server
4. Server request is processed by server code (usually a servlet)
5. Server returns data (usually a text document-html or XML) containing the result
6. XMLHttpRequest object calls the callback() function and processes the result
7. The browser document (html) is updated

10

1. Client Event

- Event
- In example below, the validate() function is called each time the user presses a key in the form field

```
<input type="text" size="20" id="userid" name="id"
       onkeyup="validate();">
```

How would you change this to validate once the user had entered data in a form field?

11

2. XMLHttpRequest Object is Created

- Validate function creates an XMLHttpRequest object

```
var req;
function validate() {
   var idField = document.getElementById("userid");
   var url = "validate?id=" + encodeURI(idField.value);
   if (typeof XMLHttpRequest != "undefined") {
      req = new XMLHttpRequest();
   } else if (window.ActiveXObject) {
      req = new ActiveXObject("Microsoft.XMLHTTP");
   }
   req.open("GET", url, true);
   req.onreadystatechange = callback;
   req.send(null);
}
```

Browser dependent code, but IE7 supports XMLHttpRequest

Notice weak typing style

Note use of function as first class object

If the call is asynchronous (true), the callback function must be specified

You can also send content

12

XMLHttpRequest Object

- Allows your code to build the request for the server
- History
  - Microsoft first implemented the XMLHttpRequest object in Internet Explorer 5 for Windows as an ActiveX object.
  - Engineers on the Mozilla project implemented a compatible native version for Mozilla 1.0 (and Netscape 7).
  - Apple has done the same starting with Safari 1.2.
  - IE7 now supports the XMLHttpRequest object
- W3C is now working on a standard that will
  - Encapsulate existing functionality
  - Possibly expand functionality

13

Common XMLHttpRequest Methods

- open("*method*", "*URL*"[, *asyncFlag*]) - Initializes the request parameters (destination URL, method, and asynchronous flag)
- send(*content*) - Transmits the request, optionally with postable string or DOM object data
- setRequestHeader("*label*", "*value*") - Assigns a label/value pair to the header to be sent with a request
- abort() - Stops the current request
- getAllResponseHeaders() - Returns complete set of headers (labels and values) as a string
- getResponseHeader("*headerLabel*") - Returns the string value of a single header label

14

Common XMLHttpRequest Properties

- onreadystatechange - Event handler (function) for an event that fires at every state change
- readyState -Object status integer:
  - 0 = uninitialized
  - 1 = loading
  - 2 = loaded
  - 3 = interactive
  - 4 = complete
- responseText - String version of data returned from server process
- responseXML - DOM-compatible document object of data returned from server process
- status - Numeric code returned by server (e.g., 404)
- statusText - String message accompanying the status code

Depends on the content-type of your response

15

3. XMLHttpRequest object calls the server

```
req.send()
```

Will cause the server to be called, using the url previously set

- If a POST method is used, a Content-Type header should be set on the XMLHttpRequest object, as in:

```
req.setRequestHeader("Content-Type",
       "application/x-www-form-urlencoded");
```


16

4. Server request is processed by server

- Server processes the XMLHttpRequest as it would any Http request

4. Server Code

```
public class ValidateServlet extends HttpServlet {
   private ServletContext context;
   private HashMap users = new HashMap();
public void init(ServletConfig config) throws
   ServletException {
   this.context = config.getServletContext();
   users.put("greg","account data");
   users.put("duke","account data"); }
public void doGet(HttpServletRequest request,
   HttpServletResponse response) throws IOException,
   ServletException {
   String targetId = request.getParameter("id");
   response.setContentType("text/xml");
   response.setHeader("Cache-Control", "no-cache");
   response.getWriter().write("<message>");
   if ((targetId != null) &&
       !users.containsKey(targetId.trim()))
       response.getWriter().write("valid");
   else response.getWriter().write("invalid");
   response.getWriter().write("</message>"); }
```

White space not needed in response

5. Server Returns a Document

- Content-type must be set to MIME type consistent with the output, for example:
  - text/xml for xml responses
  - text/json for JavaScript object responses
- Cache control header must be set to "no-cache" (keeps browsers from locally caching responses in which duplicate requests may return different responses)

```
response.setContentType("text/xml");
response.setHeader("Cache-Control", "no-cache");
```

19

6. XMLHttpRequest Object Calls callback()

Remember you set the callback function with
`req.onreadystatechange = callback;`

```
function callback() {
  if (req.readyState == 4) {
    if (req.status == 200) {
      // update the HTML DOM based on message
} } }
```

- Use standard JavaScript to modify the DOM document
- The object representation of the returned XML document is available using req.responseXML

```
function parseMessage() {
  var message =
    req.responseXML.getElementsByTagName("message")[0];
    setMessage(message.childNodes[0].nodeValue);
}
```

This is why you should set content-type to text/xml

Notice access into xml tree

20

7. The browser document is updated

- The document is re-rendered following the setting of innerHTML

```
<script type="text/javascript">
function setMessage(message) {
  mdiv = document.getElementById("userIdMessage");
  if (message == "invalid") {
    mdiv.innerHTML =
    "<div style=\"color:red\">Invalid User Id</ div>"; }
    else { mdiv.innerHTML =
      "<div style=\"color:green\">Valid User Id</ div>"; } }
</script>
<body> <div id="userIdMessage" ></div> </body>
```

How would you do this with W3C DOM?

21

Summary

- Libraries are now becoming available that build on top of the Ajax model
  - DWR
  - Google Maps
- Libraries can provide a robust API and a more natural development environment for your JavaScript and/or server code
- Ajax will likely change/improve as W3C standardization efforts continue

22

Have You Satisfied the Learning Goals?

- Understand the architecture of Ajax
- Understand the XMLHttpRequest object
- Understand how to develop an Ajax application

23