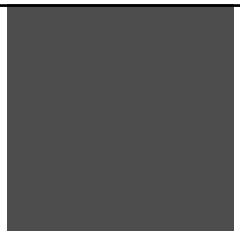


Web Application Development: JSON and Ajax Frameworks

Week 11

SWINBURNE UNIVERSITY OF TECHNOLOGY

Content

- JSON
- Ajax and JSON
- Client Side Ajax Frameworks
- Server Side Ajax Frameworks

2

JSON

- JSON stands for “JavaScript Object Notation”
- This is a data format that is useful for transmission between server and client
- It’s primary benefit is that it represents data as JavaScript data structures, so that once received on the client, the data can be directly manipulated by JavaScript code without any extraction or manipulation
- However, JSON parsers are not standard on most computer languages.
- Support to manipulate JSON data is limited.

3

JSON

- JSON is
  - easy for humans to read and write
  - easy for machines to parse and generate
  - a subset of JavaScript (Standard ECMA-262 3rd Edition - December 1999)
  - for data-interchange only (no variables or control structures)
- JSON is built on two structures:
  - An unordered collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
  - An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.
- See www.json.org

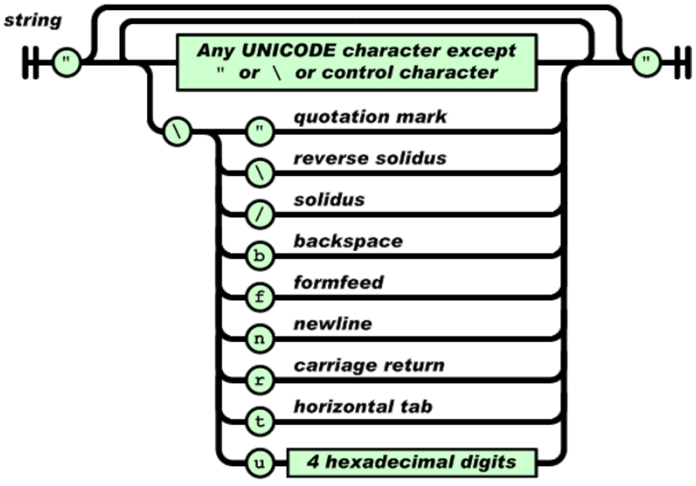
4

JSON Datatypes (JSON is Typed)

- Value
  - String (double-quoted)
    - Special characters: \“, \b, \n, \f, \r, \t, \u, \\, \/
  - Number
  - Boolean value (true/false)
  - null
  - Object
  - Array
- Object: a collection of string/value pairs
- Array: a list of values

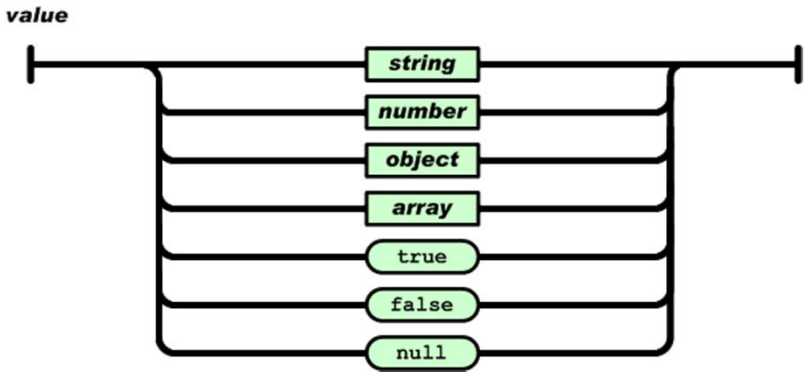
5

JSON String



6

JSON Value



7

Objects in JavaScript and in JSON

JavaScript

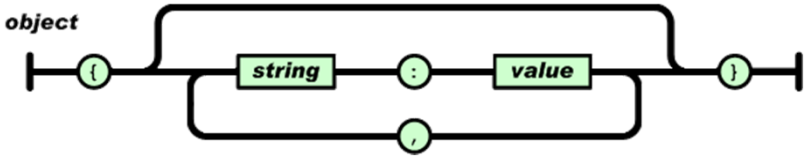
```
var member = new Object();
    member.name = "Jobo";
    member.address = "123 xyz
    Rd";
    member.isRegistered = true;
```

```
var member =
{ name: "Jobo",
  address: "123 xyz Rd",
  isRegistered: true,
};
```

JSON

```
{ "name": "Jobo",
  "address": "123 xyz Rd",
  "isRegistered": true,
}
```

Note that in JSON, the field names are strings, with quotes.



8

Arrays

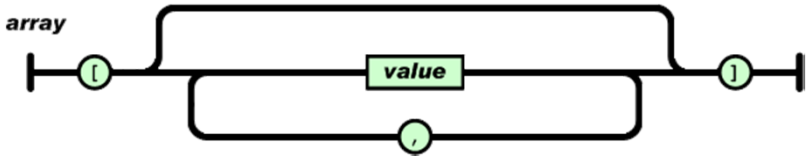
JScript

```
var myArray = new Array(1800, “Princes”, “Hwy”);
```

```
var myArray = [1800, “Princes”, “Hwy”];
```

JSON

```
[1800, “Princes”, “Hwy”]
```



9

Objects and Arrays

```
{“members”: [
   { “name”: “Jobo”,
     “address”: “123 xyz Rd”,
     “isRegistered”: true,
   },
   { “name”: “Frodo”,
     “address”: “987 pqr Rd”,
     “isRegistered”: false,
   }
 ]
}
```

This is a more complex object.

We have a single string/value pair, where the string is “members”, and the value is an array of two collections of three string/value pairs.

10

JSON is JavaScript

- Since JSON is a subset of JavaScript, it can be used in JavaScript on the client with no problems. For example:

```
var myJSONObject = {“members”: [
    { “name”: “Jobo”,
      “address”: “123 xyz Rd”,
      “isRegistered”: true,
    },
    { “name”: “Frodo”,
      “address”: “987 pqr Rd”,
      “isRegistered”: false,
    }
 ]
}
```

11

JSON is JavaScript

- Another example:

```
var myJSONObject = {
  "bindings": [
    {"ircEvent": "PRIVMSG", "method": "newURI", "regex": "^http://.*"},
    {"ircEvent": "PRIVMSG", "method": "deleteURI", "regex": "^delete.*"},
    {"ircEvent": "PRIVMSG", "method": "randomURI", "regex": "^random.*"}
  ]
};
```

- Members can be retrieved using dot or subscript operators.

```
myJSONObject.bindings[2].method    // “randomURI "
```

12

Conversion between JSON and JavaScript

- To parse a JSON string as JavaScript we can use the eval() function

```
myMembers = eval({"members": [
  { "name": "Jobo",
    "address": "123 xyz Rd",
    "isRegistered": true,
   },
  { "name": "Frodo",
    "address": "987 pqr Rd",
    "isRegistered": false,
   }]
});
alert (myMembers.members[1].address);
```

13

Conversion between JSON and JavaScript

- JSON is typically retrieved on the client as the responseText property of an XHR object. It is thus received as a string

```
var myJSONText = xhr.responseText;
var myObject = eval('(' + myJSONText + ')');
```

- WARNING : 'eval' can compile and execute any JavaScript program so there can be security issues – ie, malicious JavaScript code may have been sent from the server (rather than simply data as JSON objects)

14

Conversion between JSON and JavaScript

- So, use the parseJSON() function if security is a concern

```
var myObject = myJSONText.parseJSON();
```

- Within JavaScript we can convert a JavaScript object/array to JSON text by using the toJSONString() function

```
myJSONText = myObject.toJSONString();
```

15

JSON vs XML

- JSON is an alternative to XML for representing structured data to be retrieved from an application server and manipulated on a client
- JSON is text, and is retrieved as such (by using responseText)
- Once accessed on the client, JavaScript functions can directly manipulate the information, rather than using the DOM API, or XSLT transformation
- Often the JSON representation of data will be (a little) shorter than XML representation

16

XML

```
<!DOCTYPE glossary PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
<glossary>
 <title>example glossary</title>
 <GlossDiv>
  <title>S</title>
  <GlossList>
   <GlossEntry ID="SGML" SortAs="SGML">
    <GlossTerm>Standard Generalized Markup Language</GlossTerm>
    <Acronym>SGML</Acronym>
    <Abbrev>ISO 8879:1986</Abbrev>
    <GlossDef>
     <para>A meta-markup language, used to create markuplanguages such as DocBook.
     </para>
     <GlossSeeAlso OtherTerm="GML"/>
     <GlossSeeAlso OtherTerm="XML"/>
    </GlossDef>
    <GlossSee OtherTerm="markup"/>
   </GlossEntry>
  </GlossList>
 </GlossDiv>
</glossary>
```

Same Data in JSON

```
{"glossary": {
  "title": "example glossary",
  "GlossDiv": {
    "title": "S",
    "GlossList": {
      "GlossEntry": {
        "ID": "SGML",
        "SortAs": "SGML",
        "GlossTerm": "Standard Generalized Markup Language",
        "Acronym": "SGML",
        "Abbrev": "ISO 8879:1986",
        "GlossDef": {
          "para": "A meta-markup language, used to create markup languages such as DocBook.",
          "GlossSeeAlso": ["GML", "XML"]
        },
        "GlossSee": "markup"
      }
    }
  }
}
}
```

Basically the same structure in the data, but rather than nested tags there are nested JSON objects, and the representation is somewhat terser.

Requesting JSON Data Directly

- If a server-side application has written data in JSON format to a text file, then this can be read as follows
- xhr.open("GET", "myJSONFile.txt", true);
  - Content-type of the server's response is "text/plain"
  - The XHR object does not understand anything about JSON!
  - Data will be received as text in xhr.responseText. It is not validated against JSON rules automatically.
- Contrast with direct reading of an XML file
- xhr.open("GET", "myXMLFile.xml", true);
  - Content-type of the server's response is "application/xml" for Apache
  - Data will be received as a DOM object in xhr.responseXML. It is validated against XML rules automatically

19

Example : Adding JSON Data to a Web Page

- Data about classes offered by a university is stored in a text file, in JSON format, on the application server
- The application presents a web page to the user, with a button; when the button is pressed, details of the courses are retrieved from the server and displayed on the screen
- On the button press, an Ajax request seeks to retrieve the text from the text file on the server, and, once retrieved by the client, is manipulated and displayed as required
- The client processes the data by making use of the knowledge of the JSON object retrieved from the server

20

Adding JSON Data to Web Page (Cont'd)

File classes.txt

```
{ "class1": [
   {
    "classID": "CS115",
    "department": "Computer Science",
    "credits": 3,
    "req": "yes",
    "instructor" : "Adams",
    "title" : "Programming Concepts"
    },
    {
    "classID": "CS205",
    "semester": "fall",
    "department": "Computer Science",
    "credits": 3,
    "req": "yes",
    "instructor" : "Dykes",
    "title" : "JavaScript"
    },
    {
    "classID": "CS255",
    "semester": "fall",
    "department": "Computer Science",
    "credits": 3,
    "req": "no",
    "instructor" : "Brunner",
    "title" : "Java"
    }
   ]
}
```

This is the data, in JSON format, stored on application server as file classes.txt

21

Adding JSON Data to Web Page (Cont'd)

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head><meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
   <title>Checking Courses</title>
   <script type="text/javascript" src="xhr.js"></script>
   <script type="text/javascript" src="jsonex.js"></script>
</head>
<body>
   <h1>Checking courses</h1>
   <form>
      <input type = "button" id="reqDoc" value = "Check courses" />
   </form>
   <script type="text/javascript">
      var myDoc = document.getElementById('reqDoc');
      myDoc.onclick = getDoc;
   </script>
   <div id="title"></div>
</body>
</html>
```

File jsonex.htm

22

Adding JSON Data to Web Page (Cont'd)

```
var xhr = createRequest();

function getDoc()
{
  if (xhr) {
     xhr.open("GET", "classes.txt", true);
     xhr.onreadystatechange = function() {
       if ((xhr.readyState == 4) && (xhr.status == 200)) {
          var jsonResp = xhr.responseText;
          jsonObj = eval("(" + jsonResp + ")"); // jsonResp is now represented by  a Javascript object
          findClass(jsonObj); //the function findClass is on the next slide
       }
     }
     xhr.send(null);
  }
}
```

File jsonex.js

23

Adding JSON Data to Web Page (Cont'd)

```
function findClass(jsonTxt)
{  for (i=0; i < jsonTxt.class1.length; i++) {
      var title = jsonTxt.class1[i].title; var req = jsonTxt.class1[i].req;
      var myEl = document.createElement('p');
      var newText = title + " is the name of a course in the Computer Science department.";
      var myTx = document.createTextNode(newText);
      myEl.appendChild(myTx);
      var course = document.getElementById('title');
      course.appendChild(myEl);
      if (req == 'yes') {
        var addlText = " This is a required course.";
        var addlText2 = document.createTextNode(addlText);
        myEl.appendChild(addlText2);
      }
      else {
        var addlText = " This is not a required course.";
        var addlText2 = document.createTextNode(addlText);
        myEl.appendChild(addlText2);
      }
   }
}
```

Here we have easy retrieval from the JSON data, with direct access using standard JavaScript, coupled with the usual messy access to the DOM necessary to build the HTML for displaying the data!!

File jsonex.js (ctd)

24

Using a JSON parser

```
function getDoc()
{ var xhr = createRequest;
  if (xhr) {
     xhr.open("GET", "classes.txt", true);
     xhr.onreadystatechange = function() {
       if ((xhr.readyState == 4) && (xhr.status == 200)) {
          var jsonResp = xhr.responseText;
          jsonObj = jsonResp.parseJSON();
          ……..
```

25

Using JSON on the Server

- The previous example simply used a text file on the server already in JSON format
- But how do we get such a file in JSON format on the server?
- Usually by manipulating data from some other source, on the server, and converting it to JSON format
- So we need a way to work with JSON on the server
- Basically JavaScript does not run on the server!!
- However, there are libraries that enable formation of JSON data from server-language data structures with most (all?) server-side languages
- We will, as usual, look at PHP

26

Using JSON with PHP

- In this example we will
  - send an Ajax GET request to a PHP file on the server.
  - The PHP file will take a small PHP array, convert it to JSON format using the PHP JSON library, and echo the resulting text to the client
  - On the client, the resulting JSON will be retrieved, and displayed
- On the server, we use the library JSON-PHP, downloaded from http://mike.teczno.com/JSON/JSON.phps and saved as JSON.php on our server

27

Using JSON with PHP : Client HTML

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head><meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
   <title>My Array</title>
   <script type="text/javascript" src="xhr.js"> </script>
   <script type="text/javascript" src="jsonex2.js"> </script>
</head>
<body onload="myRequest()">
   <div id="display"></div>
</body>
</html>
```

File jsonex2.htm

28

Using JSON with PHP : Client HTML

```
var xhr = createRequest();

function myRequest()
{
  if (window.overrideMimeType) { request.overrideMimeType("text/xml");}
  if(xhr) {
     xhr.open("GET", "array.php", true);
     xhr.onreadystatechange = function() {
        if ((xhr.readyState == 4) && (xhr.status == 200)) {
           var jsonResp = xhr.responseText;
           jsonObj = eval("(" + jsonResp + ")");
           var display1 = document.getElementById('display');
           display1.innerHTML = jsonObj;
        }
     }
     xhr.send(null);
  }
}
```

Just shows the JSON without anipulation

File jsonex2.js

29

Using JSON with PHP : Server-Side PHP

```
<?php
require_once('JSON.php');
$myJSON = new Services_JSON();

$av1 = array(1, 3, 'x');
$response = $myJSON-> encode($av1);
echo ($response);
?>
```

This is how we convert a PHP data structure to the corresponding JSON structure

File array.php

30

Ajax Frameworks

- Ref : http://en.wikipedia.org/wiki/Ajax_framework
- "An **Ajax framework** is a framework that helps to develop web applications that use *Ajax*, a collection of technologies used to build dynamic web pages on the client side. Data is read from the server or sent to the server by JavaScript requests. However, some processing at the server side may be required to handle requests, such as finding and storing the data. This is accomplished more easily with the use of a framework dedicated to process Ajax requests. The goal of the framework is to provide the Ajax engine and associated server and client-side functions."

31

Ajax Client Side Frameworks

- A client side framework is basically a Javascript file (or files) that contain the definitions of several functions
- A major such framework is jQuery: http://jquery.com/
- Also see http://en.wikipedia.org/wiki/JQuery
- We will show jQuery in action on a very simple example – basically our first Ajax Example.
- This takes care of browser differences
- The framework has many features to assist with client-side development

32

HTML

```
<!– file simpleajax.htm -->
<HTML XMLns="http://www.w3.org/1999/xHTML">
 <head>
   <title>A Simple Ajax Example</title>
   <script src="http://ajax.microsoft.com/ajax/jquery/jquery-1.9.1.min.js"
           type="text/javascript"></script>
   <script type="text/javascript" src="simpleajax.js"> </script>
 </head>
 <body> TO FOLLOW
```

33

HTML (cont'd)

```
 <body>
   <H1>Fetching data with Ajax  </H1>
   <form>
      <label>User Name:  <input type="text" name="namefield"></label>
      <label><br>Password: <input type="text" name="pwdfield"><br><br> </label>
      <input name="submit" type = "button" onClick =
           "getData('data.php', namefield.value, pwdfield.value) " value = “Fetch">
   </form>
      <div id="response">
           <!-- Our message will be echoed out here -->
      </div>
 </body>
</HTML>
```

34

JavaScript

```
// file simpleajax.js
function getData(dataSource, aName, aPwd)  {
   $.ajax({
      type: "POST",
      url: dataSource,
      data: "name="+aName+"&pwd="+aPwd,
      success: function(msg){
         $("#response").html(msg);
      }
   });
}
```

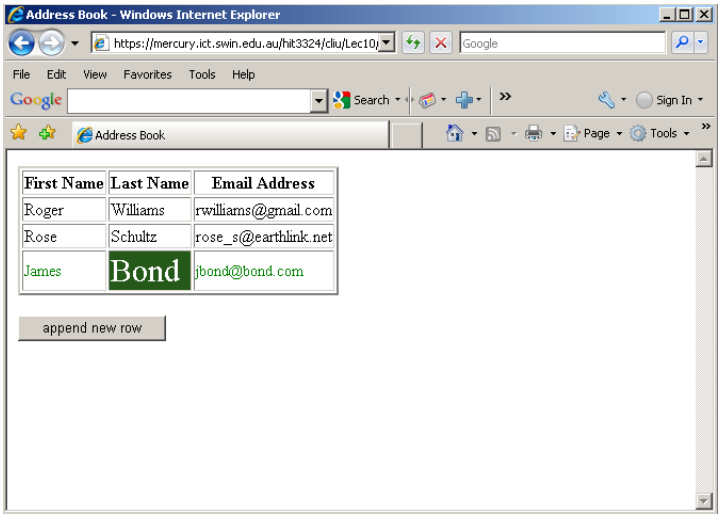
35

PHP

```
<?php
    // get name and password passed from client
    $name = $_POST['name'];
    $pwd = $_POST['pwd'];
    // write back the password concatenated to end of the name
    sleep(1);
    echo ($name." : ".$pwd)
?>
```

36

One More Example – Working on Table



37

JQueryTable.htm

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
 <title>Address Book</title>
 <script src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-1.9.0.min.js"></script>
 <script type="text/javascript">
  function addressBookItem (fname, lname, email) {
     this.fname= fname; this.lname = lname; this.email = email;
  }
  addressBookItem.prototype.write = function() {
     var adrbook = "<tr><td>" + this.fname + "</td>"; adrbook += "<td>" + this.lname + "</td>";
     adrbook += "<td>" + this.email + "</td></tr>"; document.write(adrbook);
  }
  //jquery
  $(document).ready(function() { // Handler for document.ready() called.
    $("#appendRow").click(function() {
     var firstName =prompt("Please enter the first name","");
     var lastName =prompt("Please enter the last name","");
     var email=prompt("Please enter the email address","");
     var newRow = $("<tr><td>"+firstName+"</td><td>"+lastName+"</td><td>"+email+"</td></tr>");
     $("#addressBookTbl").append(newRow);
```

38

JQueryTable.htm (Cont'd)

```
      newRow.find('td').mouseover(function() {
        $(this).css("fontSize","30px");   $(this).css("color","#ffffff");
        $(this).css("backgroundColor","#245919");
      });
      newRow.find('td').mouseout(function() {
        $(this).css("fontSize","14px"); $(this).css("color","green");
        $(this).css("backgroundColor","#ffffff");
      });
    });
  });
 </script>
</head>
<body>
 <script type="text/javascript">
   var aB1 = new addressBookItem('Roger', 'Williams', 'rwilliams@gmail.com');
   var aB2 = new addressBookItem ('Rose', 'Schultz', 'rose_s@earthlink.net');
   document.write("<table id=\"addressBookTbl\" border=\"2\"><tr><th>First Name</th><th>Last
Name</th><th>Email Address</th></tr>");
   aB1.write(); aB2.write(); document.write("</table>");
 </script><br/>
 <input type="button" id="appendRow" value="append new row"/>
</body>
</html>
```

39

Server-Side Frameworks

- An Ajax Server-Side Framework generally provides a mechanism to write the whole of an application using a mixture of HTML and a server-side language such as PHP
- The idea is that the code that would normally be written in JavaScript is actually written in the server-side language, and, when the main page is loaded, this gets automatically translated into JavaScript that is then associated with the page loaded into the browser
- Systems built with such frameworks generally do most of the processing on the server. Thus the JavaScript that is created on the client is chiefly designed to invoke the appropriate server-side functions, passing the relevant data and managing the XHR objects
- Frameworks exist for PHP, ASP.Net, Java Sever Pages, etc

40

Server-Side Frameworks

- Several server-side frameworks are listed in the article below
  - http://en.wikipedia.org/wiki/List_of_Ajax_frameworks
- Two of the more famous frameworks are
  - Google Web Toolkit (Java)
  - ASP.NET Ajax (Microsoft's .NET framework)
- We shall look at a PHP framework, SAJAX

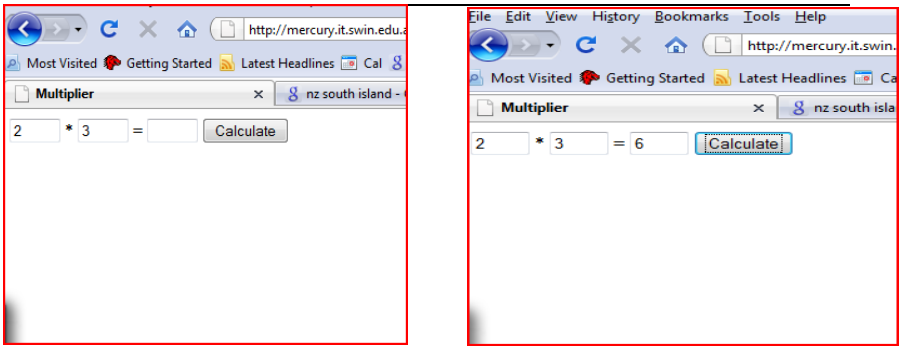
41

Example SSF - SAJAX

- We will demonstrate SAJAX, which is designed to automatically construct the client-side programs that will connect to the server in order for the server to carry out processing
- Thus we do not need to write any Javascript ourselves
- We will illustrate by implementing a simple calculator to multiply two numbers.
  - The numbers are entered in the browser
  - The user presses a button
  - The numbers are sent off to the server, where they are multiplied
  - The answer is sent back to the client
  - When it is received, it is displayed on the client
- OF COURSE THIS APPLICATION CAN BE WRITTEN AS A JAVASCRIPT PROGRAM ON THE CLIENT WITH NO SERVER NEEDED. THE POINT OF DOING IT THIS WAY IS TO ILLUSTRATE THE WAY IN WHICH SAJAX WORKS!

42

Example : A Simple interactive calculator



http://mercury.it.swin.edu.au/cos30020/cliu/Lec11/mult_ex.php
The data entered (two numbers) is sent to the server, where a PHP function multiplies the numbers and sends the result back to the client.
The whole example is in a single php file, **mult_ex.php** which has to be loaded into the browser.

43

Example : A Simple interactive calculator

```
<?
    require("Sajax.php");

    function multiply($p, $q) {
        return $p * $q;
    }

    sajax_init();
    sajax_export("multiply");
    sajax_handle_client_request();

?>
```

This is the the start of the URL that has to be invoked

This is how Sajax is made available

This is the php function that will run on the server

This is boiler-plate code. In this example, the function to export is “multiply”

44

Example : A Simple interactive calculator

```
<html>
<head>
    <title>Multiplier</title>
    <script>
    <?
    sajax_show_javascript();
    ?>

    function show_results(r) {
        document.getElementById("r").value = r;
    }

    function do_multiply() {
        var p, q;

        p = document.getElementById("p").value;
        q = document.getElementById("q").value;
        x_multiply(p, q, show_results);
    }
    </script>

</head>
```

This is the the start of the HTML part of the main file

This is how the Sajax-created JavaScript is made available

This writes result in correct place. It is the Sajax-generated XHR call-back function

This function is called when the button on the UI is pressed

This is a Sajax-created function that calls "multiply" on the server. It encapsulates the creation and use of an XHR object. Parameters are the data to be sent to the server, and the XHR call-back function.

45

Example : A Simple interactive calculator

```
<body>
    <input type="text" name= " p" id= " p" value="2" size="3">
    *
    <input type="text" name= " q" id="q" value="3" size="3">
    =
    <input type="text" name= " r" id="r" value="" size="3">
    <input type="button" name="check" value="Calculate"
        onclick="do_multiply(); return false;">
</body>
```

This is the HTML that presents the user-interface. The event handler for the button is the "do_multiply" function

46

What Goes on?

- Basically Javascript is created that
  - Creates the XHR object
  - Sends the appropriate data to the server via the XHR object, and calls the "multiply" function on the server
  - When the XHR object is "ready", the result of this multiplication is passed to the call-back function, the function show_results that we have defined to update the screen
- You can see the code by viewing the source when the application is in the browser
- The basic idea is to avoid writing JavaScript to manage the Ajax interactions. This all comes from the SAJAX library.
- The computation (multiplication in this example) is programmed on the server.
- The user still has to write JavaScript to handle results sent back from the server.

47

Our First Ajax Example, using SAJAX

```
<?
  require("Sajax.php");

  function concatenate($p, $q) {
          return $p." : ".$q;
  }

  sajax_init();
  sajax_export("concatenate");
  sajax_handle_client_request();

?>
```

The function that does the computation on the server

This and code on next 2 slides is all in file simpleajax.php

48

Our First Ajax Example, using SAJAX

```
<html>
<head>
  <title>Simple Ajax Test</title>
  <script>
  <?
  sajax_show_javascript();
  ?>

  function show_results(r) {
            document.getElementById("Results").value = r;
  }

  function do_concatenate() {
            var name, password;
            name = document.getElementById("Name").value;
            password = document.getElementById("Password").value;
            x_concatenate(name, password, show_results);
  }
  </script>
</head>
```

XHR readyStateChange event handler

Button press event handler

49

Our First Ajax Example, using SAJAX

```
<body>
  <input type="text" name="Name" id="Name" value="" size="12">

  <input type="text" name="Password" id="Password" value="" size="6">

  <input type="button" name="check" value="Get Results"
          onclick="do_concatenate(); return false;">

  <input type="text" name="Results" id="Results" value="" size="21">
</body>
</html>
```

Button press event handler

50