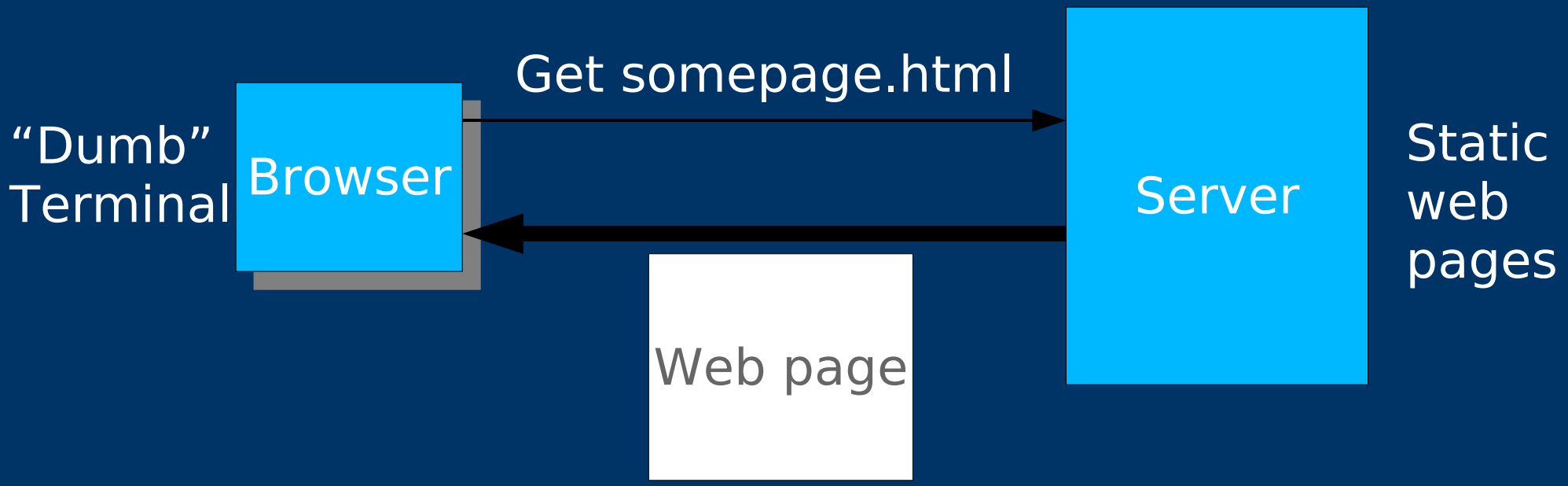
AJAX

For presentation to the
Columbia Area Linux Users Group
January 9, 2008
Eldon Ziegler
eldonz@atlanticdb.com

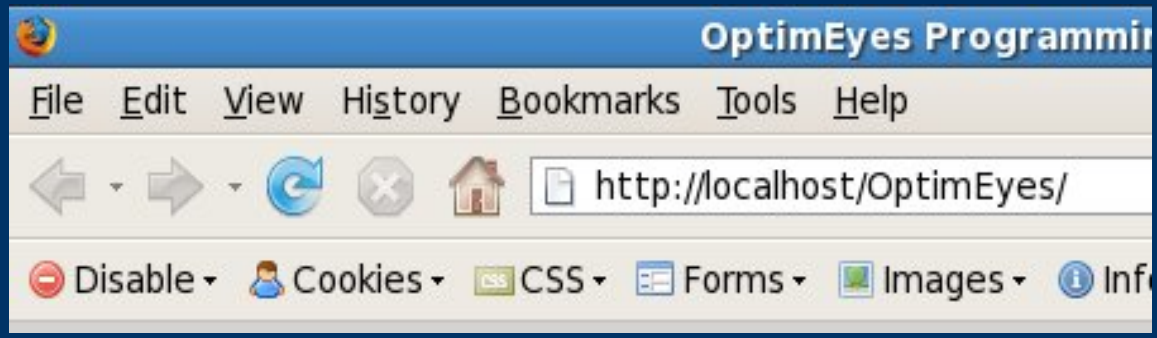
AJAX

- **A**synchronous **J**avaScript **A**nd **X**ML
  - Better web experience to users
  - Reduce bandwidth use
  - Off load processing from server to PC
  - Better software architecture

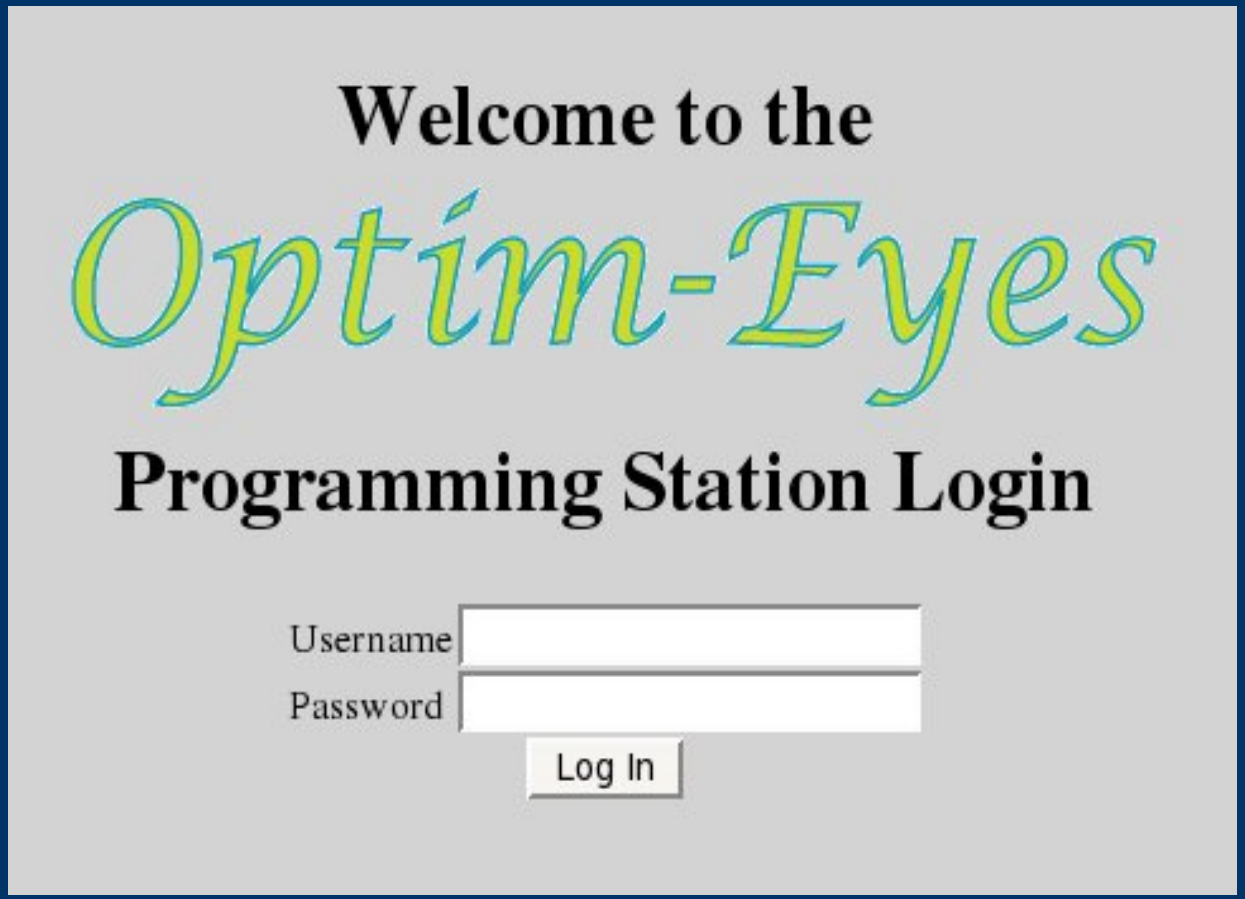
Conventional Web Communication



Sample Browser Address



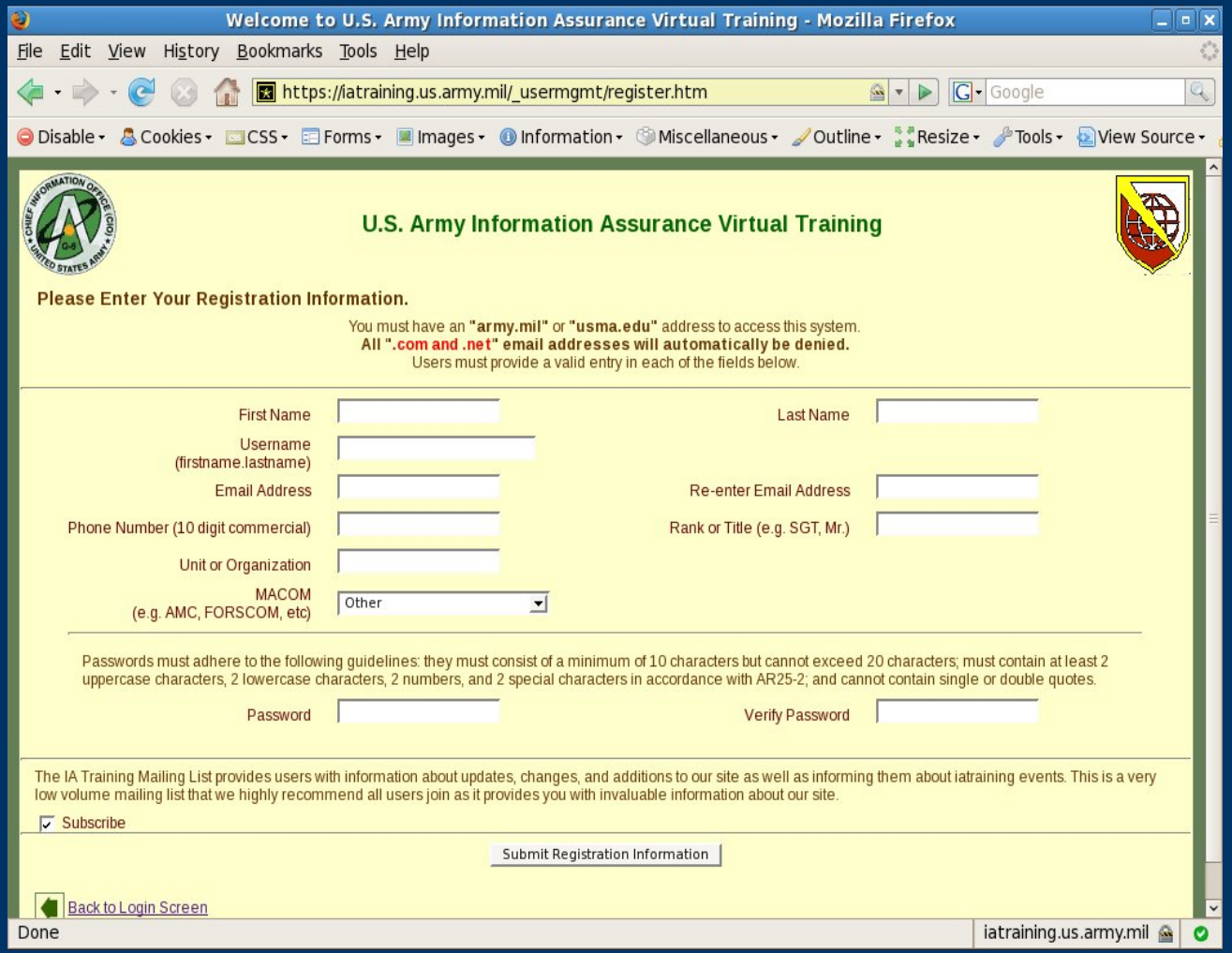
Sample Log in Page



Log in Page HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<!-- Copyright (C) 2007 Atlantic Database Systems, Inc. All rights reserved.-->
<html>
<head>
<meta name="Author" "Eldon Ziegler Atlantic Database Systems, Inc.">
<!-- <link REL="SHORTCUT ICON" href="/favicon.ico"> -->
<title>OptimEyes Programming Station Login</title>
<meta name="Keywords" content="OptimEyes, dyslexia, medical device">
<meta name="Language" content="English">
<meta name="Robots" content="All">
<script type="text/javascript" src="js/HttpClient.js"></script>
<script type="text/javascript" src="js/ajaxutilities.js"></script>
<script type="text/javascript" src="js/utilities.js"></script>

<script type="text/javascript" src="login.js"></script>
<!--
<script type="text/javascript" src="code1.js"></script>
-->

<link type="text/css" href="common.css" rel="stylesheet"/>
</head>

<body onload="OnLoginLoad()">
<h1 class="cltitle">Welcome to the<br/><img src="Optim_Logo.gif"/><br/>Programming Station Login</h1>
<table cellspacing="0" align="center">
        <tr>
                <td>
                        <div id="loginerr" class="clhidden">

                                <h2>Your login was incorrect, please try again.</h2>
                        </div>
                        <table cellspacing="0" align="center">
                                <tr>
                        <td><label for="username">Username</label></td>
                        <td><input type="text" size="20" name="username" id="username"/></td>
                        </tr>
                        <tr>

                        <td><label for="passwd">Password</label></td>
                        <td><input type="password" size="20" name="passwd" id="password"/></td>
                        </tr>
                        <tr>
                        <td colspan="2" align="center">
                                                <input type="button" name="submit" id="loginsubmit" value="Log In"
onclick="OnLoginClick()"/>
                                                <span id="loginmsg" style="display: none">One moment, please.</span>
                                        </td>
                        </tr>

                        </table>
                </td>
        </tr>
</table>
<div style="position: absolute; width:100px; height:20px;
        top: 5px; right: 5px; display:none"
    id="HttpClientStatus">Loading ...</div>
</body>
</html>
```

User Registration



Conventional Registration

User Registration with AJAX

AJAX Registration

Invoking the JavaScript

```
<head>
<title>AJAX Sample</title>
<script type="text/javascript" src="/js/HttpClient.js"></script>
<script type="text/javascript" src="/_usermgmt/register.js"></script>
<link rel="stylesheet" href="display.css" media="screen">
</head>

<body>
...
<input type="text" size="25" value="" id="username"
  name="username"                onchange="registerOnChangeName
  ()"/>
...
<div style="position: absolute; width:100px; height:20px;
         top: 5px; right: 5px; display:none"
    id="HttpClientStatus">Loading ...</div>
</body>
```

Asynchronous JavaScript

```
function registerOnChangeName ()
{
     varfirstName = Trim (GetInputText
  ('first_name'));
     varlastName = Trim (GetInputText ('last_name'));
     varusername = Trim (GetInputText ('username'));


     if (firstName && lastName && username) {
       var  xml = “<req><un>” + username +
                  “</un></req>”;
       SendToServer (“chkloginname.php”, xml,
                     regHaveChkResp);
     }
}
```

JavaScript Helper for HttpClient

```
function SendToServer (uri, xml, RespFunction)
{
     varhttpClient = new HttpClient ();
     httpClient.isAsync = true;
     httpClient.requestType = 'POST';
     httpClient.callback = RespFunction;
     httpClient.makeRequest (uri, xml,
  'text/xml');
     return httpClient;
}
```

On the Server

```
<?php
    session_start();
    session_register();

    $docinc = getenv('docinc');

    require_once $docinc . '/_p2g/ajaxutilities.php';
    require_once $docinc . '/_usermgmt/userdbhandlers.php';

    // Get and check the name.

    $xml = GetXML ();
    $name = ExtractXMLTextByTag ($xml, "un");
    $bOK = !qdbUserNameExists ($name);
```

Format the Response

```
$response = "<resp>";
$response .= "<res>" . $bOK ? "ack" : "nak" . "</res>";

if (!$bOK) {
    // Find available names.
    $nNames = 0;
    for ($i = 1; $i < 1000 && $nNames < 2; $i++) {
        $try = $name . $i;
        if (!qdbUserNameExists ($try)) {
            $response .= "<un>" . rawurlencode ($try) . "</un>";
            $nNames++;
        }
    }
}

$response .= "</resp>";
```

Send Response to Browser

```
    // Send the xml response.

    header ('Content-Type: text/xml');
    echo $response;
?>
```

JavaScript to Check User Name

```
function regHaveChkResp (responseText, responseXML)
{
  var   res = getxmlitem (responseXML, “res”);

  if (res != “ack”) {
     varsuggestions = responseXML.getElementsByTagName (“un”);
     var names = “”
     for (var i = 0; i < suggestions.length; i++) {
        if (i > 0)
           names += “, “;
        names += suggestions [i];
     }
     varerrmsg = “<span>Error: Username is already taken.<br/>
        Possibilities include: “ + names + “</span>”;
     var el = document.getElementById (“regerror”);
     el.innerHTML = errmsg;
     el.style.display = “inline”;
  }
}
```

JavaScript Uses

- Check user entered data
  - Nothing omitted
  - Correct format
  - Correct common errors
    - Blanks at the ends of log in names
- Format the display from basic data
- Display relevant data
- Communicate with server

AJAX Registration

Today's PC

Compaq - Presario E2140 Desktop
$454.99

Intel® Pentium® Dual-Core processor E2140; 1GB DDR2 SDRAM; DL DVD±RW/CD-RW drive; LightScribe labeling; 320GB hard drive

Off Load Processing

Software Architecture

Browser

- HTML for static display
- CSS for formatting
- JavaScript for dynamic interaction

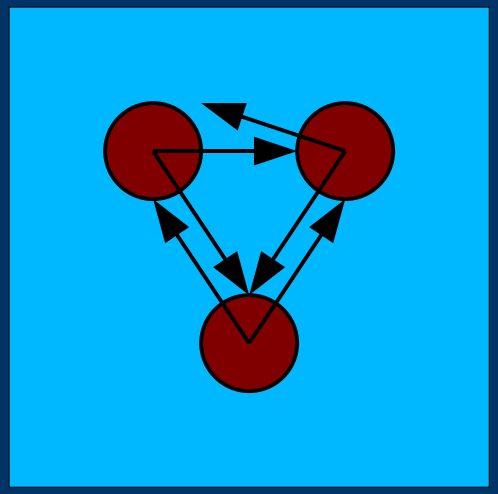
Server

- Server (PHP) for database interface

Processing

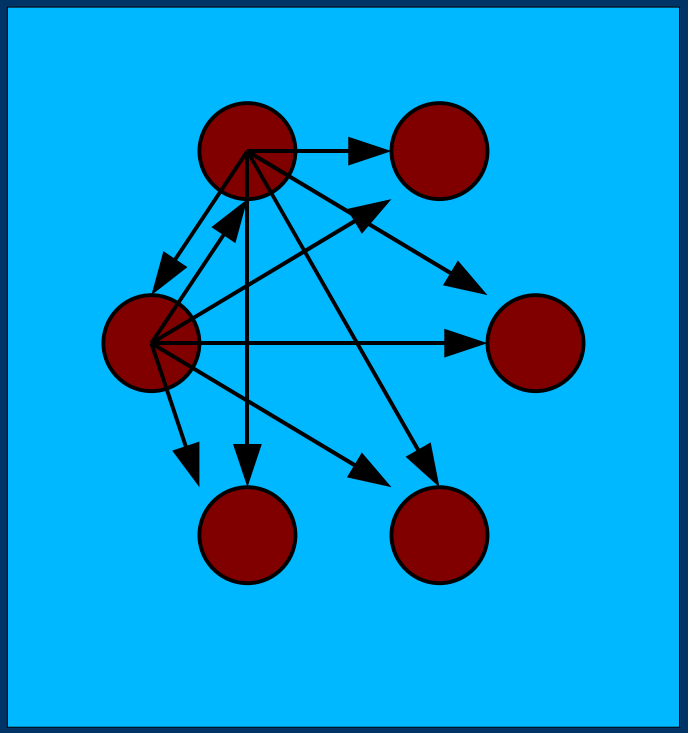
Complexity – Factorial Explosion

Small System



3! = 6 interactions

Bigger System



6! = 720 interactions

- Size of system doubled
- Complexity increased by 120 : 1

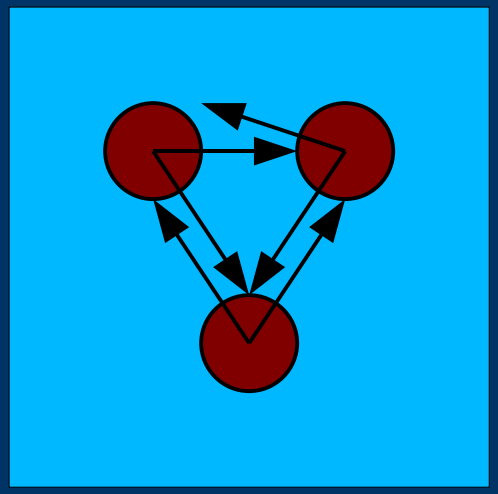
“Complexity Kills”

“Complexity kills. It sucks the life out of developers, it makes products difficult to plan, build, and test, it introduces security challenges and it causes end-user and administrator frustration.” -- Ray Ozzie, CTO Microsoft

“Open Standards and Security”
David A. Wheeler July 12, 2006

Complexity – Modular Improvement

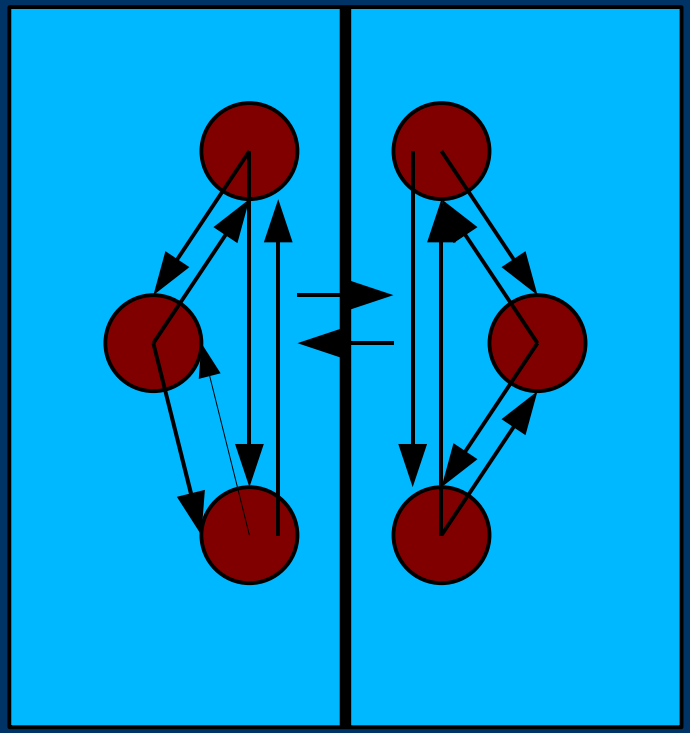
Small System



3! = 6 interactions

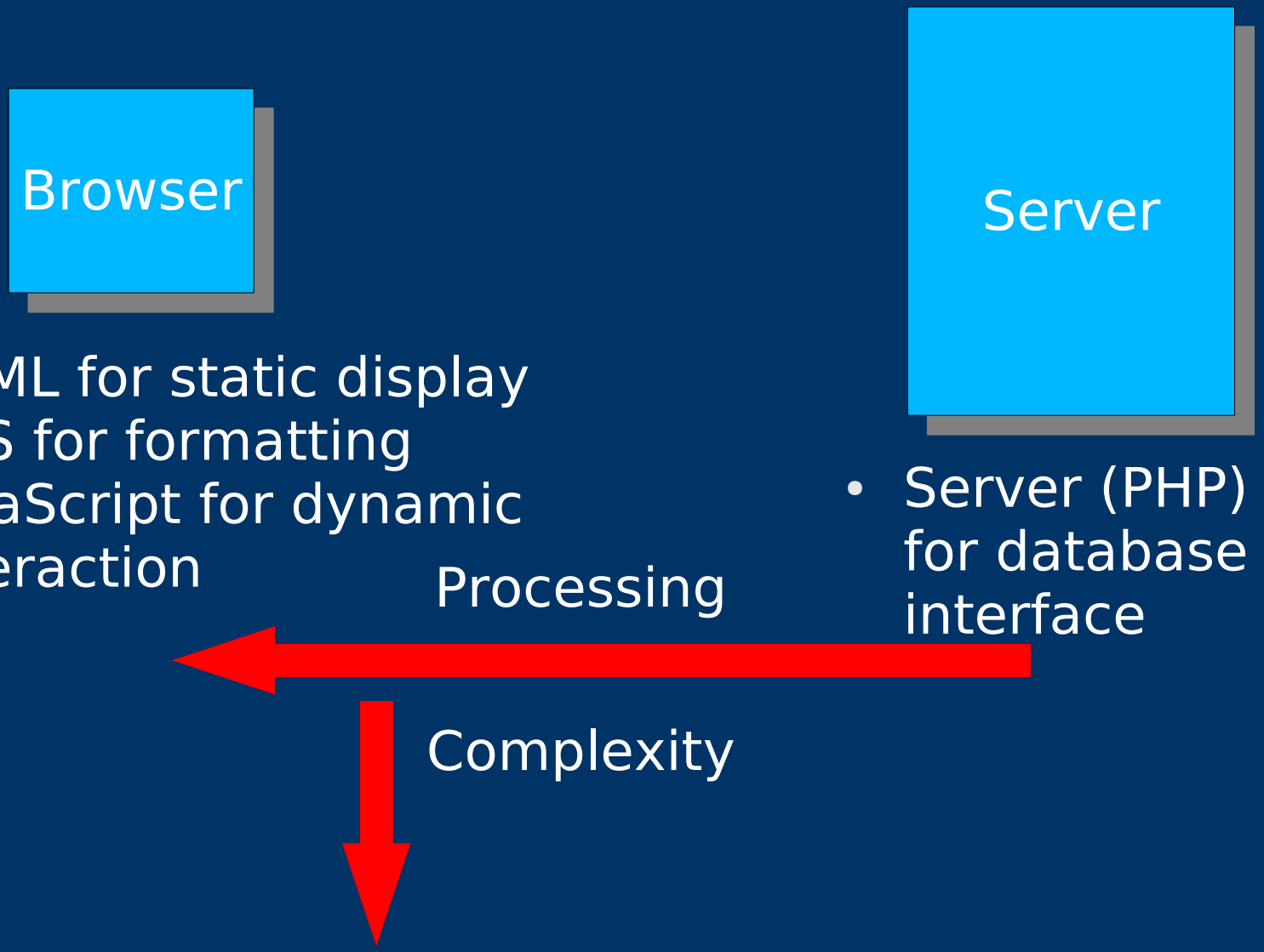
- Modular system reduced complexity by 50 : 1

Bigger System



6! = 720 interactions
2 * 3! + 2 = 14 interactions

Software Architecture



Complexity

- People problem – not computer problem
- Computer will run correct code correctly even if it's old “spaghetti” code loaded with goto's
- People have trouble keeping track of more than a few things at one time
  - Manager can handle no more that 7 things

AJAX Role

- Divide functions between server and browser
- Make clear difference between static and dynamic information
  - Static HTML
  - Dynamic changes in JavaScript
  - Limit server code functionality
    - Database interface
    - Real-time interfaces

AJAX Problems

- Inconsistent brower compliance
  - IE
- Can break Back button
  - Work-arounds have problems with IE
- Special handling for bookmarks
- Compatibility with screen readers
- Working with DOM can be challenging
- JavaScript function “scoping”

Resources

- “Understanding AJAX”, Joshua Eichorn, Prentice Hall
- “JavaScript The Definitive Guide”, David Flanagan, O'Reilly
- “Securing Ajax Applications”, Christopher Wells, O'Reilly
- “Advanced AJAX”, Shawn M. Lauriat, Prentice Hall
- Wikipedia, http://en.wikipedia.org/wiki/Ajax_%28programming%29

References

- “Ajax: A New Approach to Web Applications”, Jesse James Garrett, www.adaptivepath.com/ideas/essays/archives/000385.php
- “A List Apart”, http://alistapart.com/
- “AJAX: How to Handle Bookmarks and Back Buttons”, www.onjava.com/pub/a/onjava/2005/10/26/ajax-handling-bookmarks-and-back-button.html?page=1
- W3 Schools, www.w3schools.com

JavaScript Helper for HttpClient

```
function SendToServer (uri, xml, RespFunction)
{
     varhttpClient = new HttpClient ();
     httpClient.isAsync = true;
     httpClient.requestType = 'POST';
     httpClient.callback = RespFunction;
     httpClient.makeRequest (uri, xml,
  'text/xml');
     return httpClient;
}
```

HttpClient.js

```
function HttpClient() { }

HttpClient.prototype = {
    // type GET,POST passed to open
    requestType:'GET',

    // when set to true async calls are made
    isAsync:false,

    // where an XMLHttpRequest instance is stored
    xmlhttp:false,

    // what is called when a successful async call is
made
    callback:false,
```

Helper Functions

```
	// what is called when send is called on XMLHttpRequest
	// set your own function to onSend to have a custom loading effect
	onSend:function() {
		document.getElementById('HttpClientStatus').style.display =
'block';
	},

	// what is called when when readyState 4 is reached, this is called
before your callback
	onLoad:function() {
		document.getElementById('HttpClientStatus').style.display =
'none';
	},

	// what is called when an http error happens
	onError:function(error) {
		alert(error.message);
	},
```

Initializing HttpClient

```
// method to initialize an xmlhttpclient
init:function() {
      try {
         // Mozilla / Safari
         this.xmlhttp = new XMLHttpRequest();
      } catch (e) {
           See next page
      }
},
```

Living With IE

```
// IE
var XMLHTTP_IDS = new Array(
      'MSXML2.XMLHTTP.5.0',
      'MSXML2.XMLHTTP.4.0',
      'MSXML2.XMLHTTP.3.0',
      'MSXML2.XMLHTTP',
      'Microsoft.XMLHTTP' );
var success = false;
for (var i=0;i < XMLHTTP_IDS.length && !success; i++) {
      try {
            this.xmlhttp = new
ActiveXObject(XMLHTTP_IDS[i]);
            success = true;
      } catch (e) {}
}
if (!success) {
      throw new Error('Unable to create XMLHttpRequest.');
}
```

```
// method to make a page request
// @param string url  The page to make the request too
// @param string payload  What your sending if this is a POST request

makeRequest: function (url, payload) {
      if (!this.xmlhttp) {
            this.init();
      }
      this.xmlhttp.open (this.requestType, url, this.isAsync);

      // set onreadystatechange here since it will be reset after
            a completed call in Mozilla
      var self = this;
      this.xmlhttp.onreadystatechange = function()
                                    { self._readyStateChangeCallback(); }

      this.xmlhttp.send(payload);

      if (!this.isAsync) {
            return this.xmlhttp.responseText;
      }
},
```

```
    // internal method used to handle ready state changes

    _readyStateChangeCallback:function () {
        switch (this.xmlhttp.readyState) {
            case 2:
                this.onSend();
                break;
            case 4:
                this.onLoad();
                if  (this.xmlhttp.status == 200) {
                    this.callback (this.xmlhttp.responseText,
                                   this.xmlhttp.responseXML);
                } else {
                    this.onError (new Error ('HTTP Error Making Request:
['
                                             + this.xmlhttp.status + '] '
                                             + this.xmlhttp.statusText));
                }
            break;
        }
    }
} // End of prototype
```