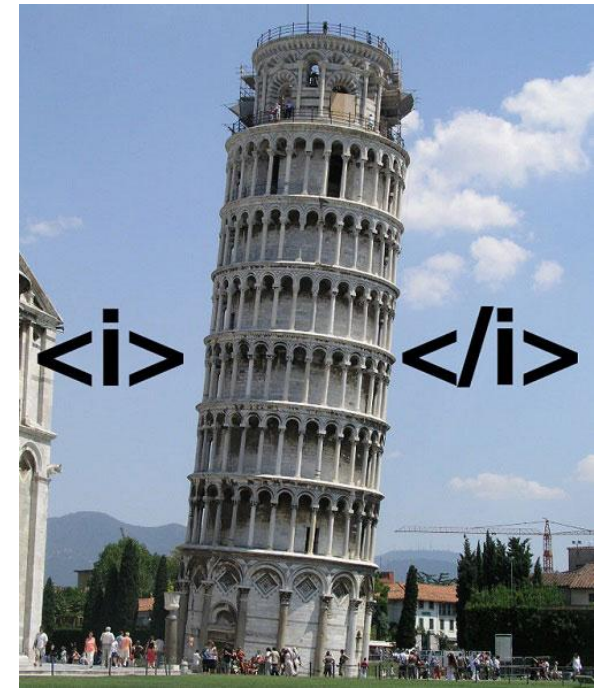


# Web Service Protocols



# HTTP: GET, POST, & Friends

# GET Requests

- Recall: http offers
  - GET, POST, PUT, DELETE
  - ...plus several more
- Request modification through key/value pairs
  - ?
  - &
- Client sends:

`http://acme.com/srv ? mybasket=6570616275 & article=656e44204456`

see REST later!

# Request Parameters: How Passed?

- GET parameters: URL text

- Can be cached, bookmarked
- Reload / back in history harmless
- Data visible in URL

GET srv?k1=v1&k2=v2 HTTP/1.1

- POST parameters: HTTP message body

- Not cached, bookmarked
- Reload / back in history **re-submits**
- Data not visible,  
not in history,  
not in server logs

POST srv HTTP/1.1

k1=v1&k2=v2

[http://www.w3schools.com/tags/ref\\_httpmethods.asp](http://www.w3schools.com/tags/ref_httpmethods.asp)

# AJAX

## (Asynchronous Javascript and XML)

# History

- Challenge: want **more interactivity** than "click link / reload *complete page*"
  - HTML's `iframes`
- Microsoft IE5 XMLHttpRequest object
  - Outlook Web Access, supplied with Exchange Server 2000
- 2005: term "AJAX" coined by Jesse James Garnett
- made popular in 2005 by Google Suggest
  - start typing into Google's search box → list of suggestions

# AJAX

- **AJAX = Asynchronous Javascript and XML**
- **web development technique for creating more interactive web applications**
  - Goal: increase interactivity, speed, functionality, usability
  - not complete page reload → small data loads → more responsive
- **asynchronous: c/s communication independent from normal page loading**
  - JavaScript
  - XML
  - any server-side PL

# Constituent Technologies

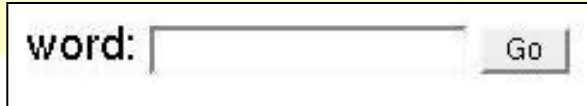
- The core: JavaScript **XMLHttpRequest** object
  - Sends data, waits for response via event handler
  - Replaces <FORM> and HTTP GET / POST
- **Client DOM manipulated** to dynamically display & interact
  - Inject response into any place(s) of DOM tree
  - client-side scripting language: JavaScript, Jscript, ...
- Some data format
  - XML, JSON, HTML, text, ...
- Some server agent
  - Servlet, script, ...



# Ajax Example: Traditional Style

- Client:

```
<form method='GET' action='http://.../ajax-ex.php'>
  word:
  <input name='wordKey' type='text'>
  <input type='submit' value='Go'>
</form>
```



word:



- Server:

```
<?
  echo 'You have entered ' . $_GET['wordKey']
    . ' and your IP is: ' . $_SERVER['REMOTE_ADDR'];
?>
```



- Client, after **page reload**: You have entered Moribundus, and your IP is: 127.0.0.1

# Step 1: Avoid Complete Page Reload

```
<form name='wordForm'>
  word:
  <input name='wordKey' type='text'>
  <input type='button' value='Go' onClick='JavaScript:callback()'>
  <div id='result'></div>
</form>
```

```
function callback()
{
  var SERVICE = 'http://.../ajax-ex.php';
  var req = new XMLHttpRequest();
  var val = document.forms['wordForm'].wordKey.value;
  req.open( 'GET', SERVICE+'?wordKey='+val, true );
  req.setRequestHeader( 'Content-Type',
                       'application/x-www-form-urlencoded' );
  req.send( null );
  req.onreadystatechange = function()
  {
    if (req.readyState == 4)
      document.forms['wordForm'].result.innerHTML =
        req.responseText;
  }
}
```

- 0 request not initialized
- 1 request set up
- 2 request sent
- 3 request in process
- 4 request complete

word: \_\_\_\_\_

You have entered Moribundus, and your IP is: 127.0.0.1

# Step 2: Avoid SUBMIT Button

- Before: just re-implemented submit; now: allow c/s activity at **any time**
  - Event handlers
- Ex: suggest keywords with every char typed
  - No submit button!

```
<input name='wordKey' onKeyUp='JavaScript:callBack()' >
```

```
<? ...
$query = "select entry from Airports
         where entry like '" . $_GET['wordKey'] . "%'";
$result = mysql_query( $query );
while ( $row = mysql_fetch_array( $result ) )
{
    print $row[ 'entry' ] . ",";
}
?>
```

*How to ship back  
& inject data?*

# JSON

- JSON = **JavaScript Object Notation**
  - Lightweight data interchange format
  - MIME type: `application/json` (RFC 4627)
  - text-based, human-readable
- alternative to XML use
  - Subset of JavaScript's object literal notation
  - 10x faster than XML parsing
  - way easier to handle
  - JSON parsing / generating code readily available for many languages

*"JSON is XML without garbage"*

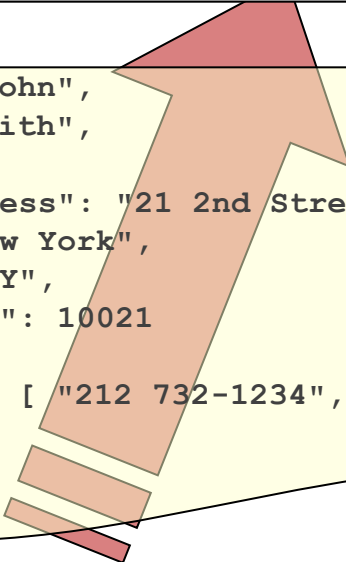
# JSON Example

- Server sends:

```
req.onreadystatechange=function()
{
  if(req.readyState==4)
  {
    var p = eval( "(" + req.responseText + ")" );
    document.myForm.firstName.value = p.firstName;
  }
}
```

- JSON string sent from server:

```
{
  "firstName": "John",
  "lastName": "Smith",
  "address":
  {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
  },
  "phoneNumbers": [ "212 732-1234", "646 123-4567" ]
}
```



- response parsing code:

```
<? echo '{' + '"firstName":' + obj.firstName + ',' +
      + '"lastName":' + obj.lastName + ',' +
      ... + '}'
?>
```

# JSON Security Concerns

- JavaScript `eval ()`
  - most JSON-formatted text is also **syntactically legal JavaScript code!**
  - built-in JavaScript `eval ()` function **executes** code received
- **Invitation to hack:**  
embed rogue JavaScript code (server-side attack),  
intercept JSON data evaluation (client-side attack)
  - **Safe alternative:** `parseJSON ()` method,  
see ECMAScript v4 and [www.json.org/json.js](http://www.json.org/json.js)
- Cross-site request forgery
  - malicious page can request & obtain JSON data belonging to another site

# AJAX / JSON Portability

- AJAX uses **standardized components**, supported by all major browsers:
  - JavaScript, XML, HTML, CSS
- XMLHttpRequest object part of **std DOM**
  - Windows: ActiveX control Msxml2.XMLHTTP (IE5), Microsoft.XMLHTTP (IE6)
- ...similarly for JSON

# Appraisal: AJAX Advantages

- Reduced **bandwidth usage**
  - No complete reload/redraw, HTML generated locally, only actual data transferred  
→ payload coming down much smaller in size
  - Can load stubs of event handlers, then functions on the fly
  
- **Separation** of data, format, style, and function
  - encourages programmers to clearly separate methods & formats:
 

Raw data / content	→ normally embedded in XML
webpage	→ HTML / XHTML
web page style elements	→ CSS
Functionality	→ JavaScript + XMLHttpRequest + server code



# Appraisal: AJAX Disadvantages

- **Browser integration**
  - dynamically created page not registered in browser history
  - bookmarks
- **Search engine optimization**
  - Indexing of Ajax page contents?
  - (not specific to Ajax, same issue with all dynamic data sites)
- **Web analytics**
  - Tracking of accessing page vs portion of page vs click?
- **Response time concerns from network latency**
  - Web transfer hidden → effects from delays sometimes difficult to understand for users
- **Reliance on JavaScript**
  - JavaScript compatibility issue → blows up code; Remedy: libraries such as `prototype`
  - IDE support used to be poor, changing
  - Can switch off JavaScript in my browser
- **Security**
  - Can fiddle with data getting into browser

# Summary

- AJAX allows to add desktop flavour to web apps
  - JSON as lightweight, fast alternative to XML
- Web programming paradigm based on existing, available standards
- Issues: browser compatibility, security, web dynamics
- Many usages:
  - real-time form data validation; autocompletion; bg load on demand; sophisticated user interface controls and effects (**trees**, menus, data tables, rich text editors, calendars, progress bars, ...); partial submit; mashups (app mixing); desktop-like web app



# Resources

- Books:
  - Michael Mahemoff: Ajax Design Patterns. O'Reilly, 2006
  - Mark Pruet: Ajax and Web Services. O'Reilly, 2006
  
- Web:
  - [www.openajaxalliance.org/](http://www.openajaxalliance.org/)
  - [w3schools.org/ajax](http://w3schools.org/ajax)
  - Mozilla Developer Center: AJAX:Getting Started
    - [developer.mozilla.org/en/docs/AJAX:Getting\\_Started](http://developer.mozilla.org/en/docs/AJAX:Getting_Started)
  - [www.json.org](http://www.json.org)

# Tool Support: Examples

- jQuery, <http://jquery.com/>

```
$( "button.continue" ).html( "Next Step..." )
```

- AJAX:

```
$.ajax({  
  url: "/api/getWeather",  
  data: {  
    zipcode: 97201  
  },  
  success: function( data ) {  
    $( "#weather-temp" ).html( "<b>" + data + "</b> degrees" );  
  }  
});
```