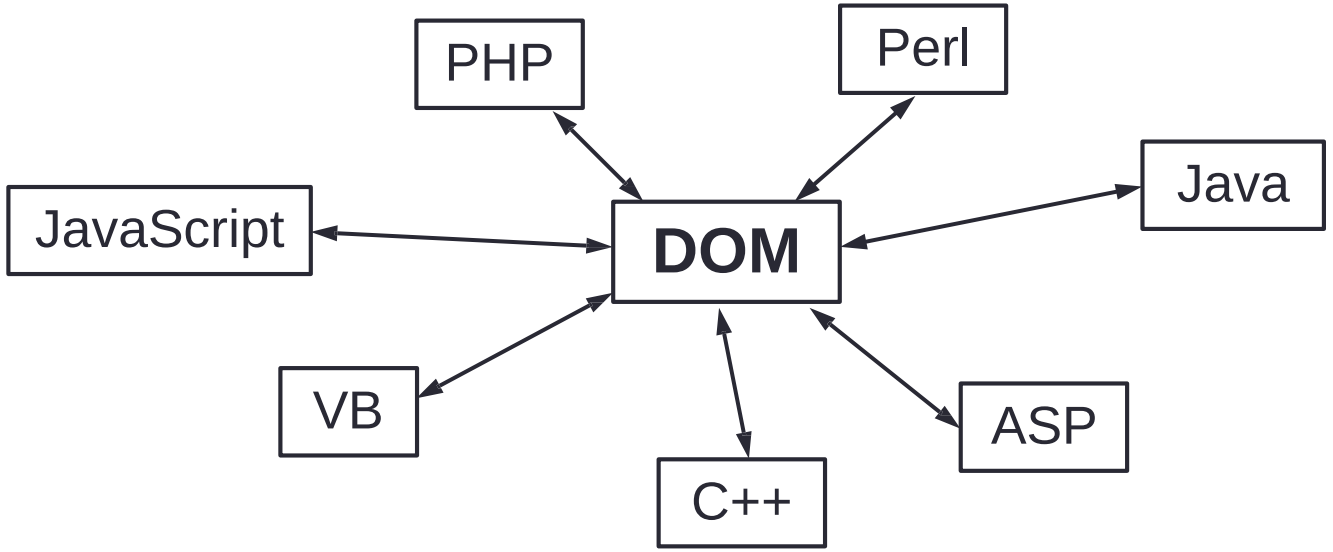
SERVICE ORIENTED WEB APPLICATIONS

XML PROCESSING WITH DOM AND SAX

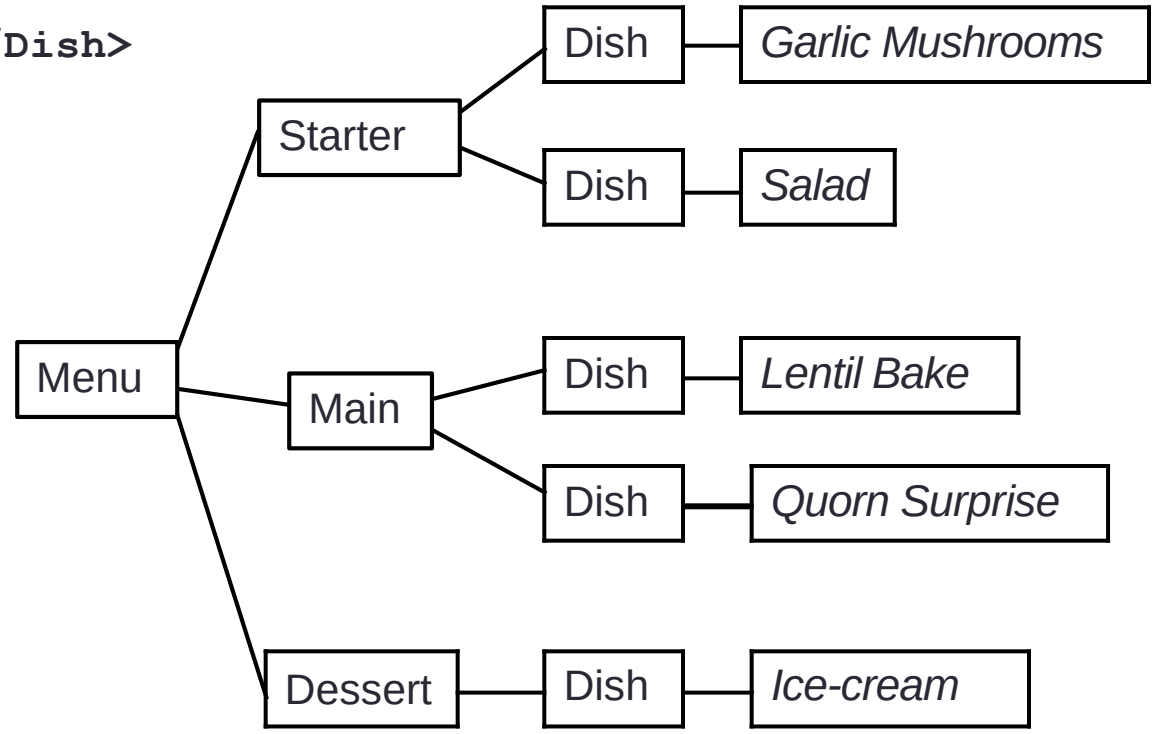
Document Object Model

- We have already seen the Document Object Model (DOM) in the context of JavaScript and HTML
- DOM is key to "advanced" XML programming
  - It allows you to manipulate XML documents in ways that are not possible using XSL
- W3C DOM is language and platform neutral



```
<?xml version="1.0" ?>
<Menu meal="lunch">
   <Starter>
      <Dish>Garlic Mushrooms</Dish>
      <Dish diet="vegan">Salad</Dish>
   </Starter>
   <Main>
      <Dish diet="vegan">Lentil Bake</Dish>
      <Dish diet="vegetarian">Quorn Surprise</Dish>
   </Main>
   <Dessert>
      <Dish>Ice-cream</Dish>
   </Dessert>
</Menu>
```

XML syntax rules mean that well formed XML documents can be represented as a tree of objects



The DOM presents a similar (although slightly more complex) view of an XML document

What is the DOM?

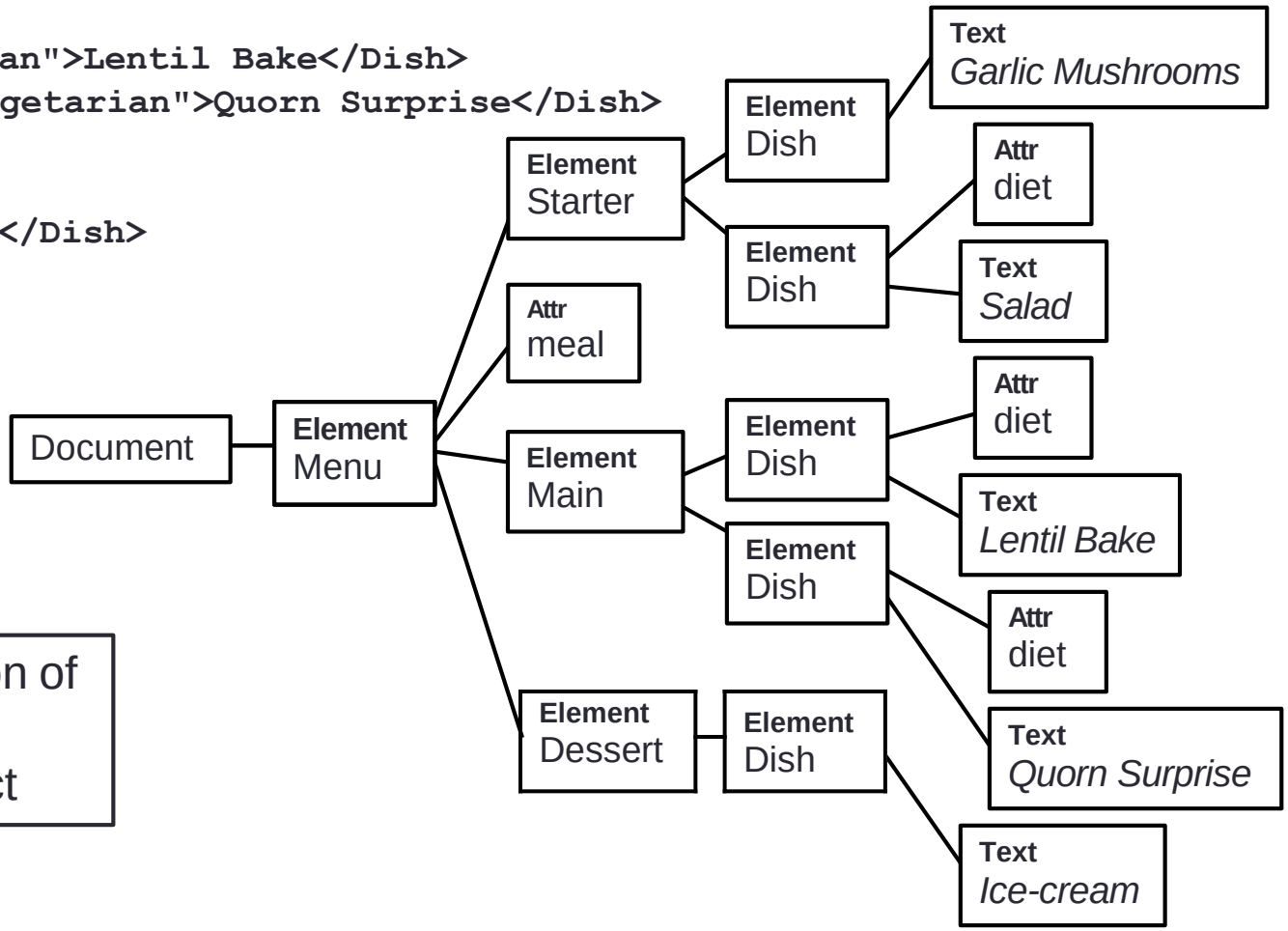
- A set of objects types that are used to represent the object tree view of an XML document
- The objects described in the DOM allow the programmer to read, search, modify, add to and delete from an XML document
- The DOM provides a standard definition of functionality for document navigation and manipulation
- A DOM API can be implemented in any programming language
  - such implementations are called **bindings**
- The W3C DOM Level 2 includes standards for the core object model, views, events, style, traversal and range

DOM Object Types

- DOM Level 1 includes 18 object types, e.g.
  - **Node** - a general type of object - everything is a node as well as belonging to one of the more specialised types
  - **Document** - to represent the whole document
  - **Element** - to represent each element (i.e. tag)
  - **Attr -** to represent attributes
  - **Text** - a node that represent the textual content of an element
  - **NodeList** - a list of nodes, e.g. all the elements that are "child" elements of another node.
- Each object type has a defined set of properties and methods

Menu example redrawn as a DOM tree

```
<?xml version="1.0" ?>
<Menu meal="lunch">
   <Starter>
      <Dish>Garlic Mushrooms</Dish>
      <Dish diet="vegan">Salad</Dish>
   </Starter>
   <Main>
      <Dish diet="vegan">Lentil Bake</Dish>
      <Dish diet ="vegetarian">Quorn Surprise</Dish>
   </Main>
   <Dessert>
      <Dish>Ice-cream</Dish>
   </Dessert>
</Menu>
```



Note the addition of a top-level document object

Draw a DOM object tree to represent the following XML document - indicate the DOM object types as shown in the previous slide

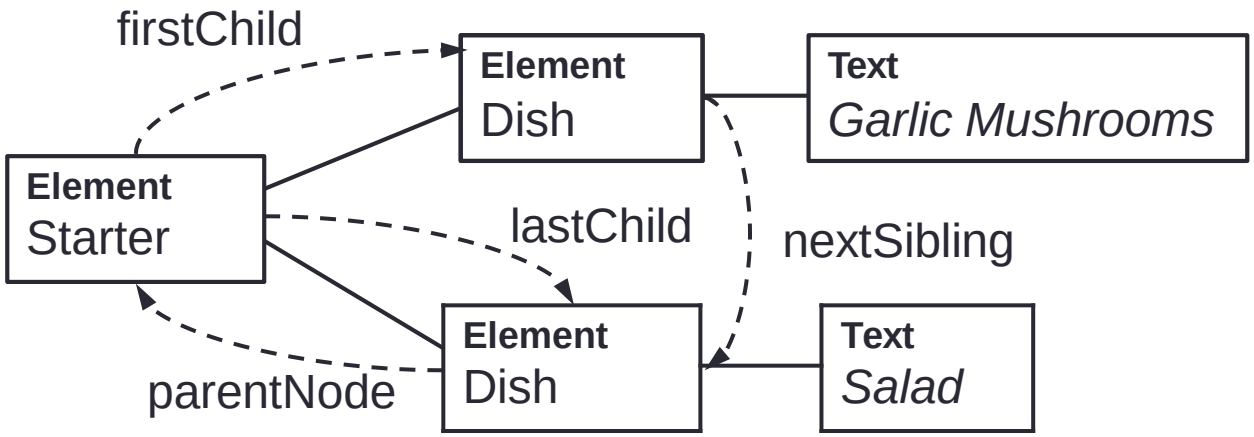
```
<?xml version = "1.0"?>
<Company>
   <Employee>
      <FirstName>Tanmay</FirstName>
      <LastName>Patil</LastName>
      <ContactNo>1234567890</ContactNo>
      <Email>tanmaypatil@xyz.com</Email>
      <Address>
         <City>Bangalore</City>
         <State>Karnataka</State>
         <Zip>560212</Zip>
      </Address>
   </Employee>
</Company>
```

So what does this give you as a programmer?

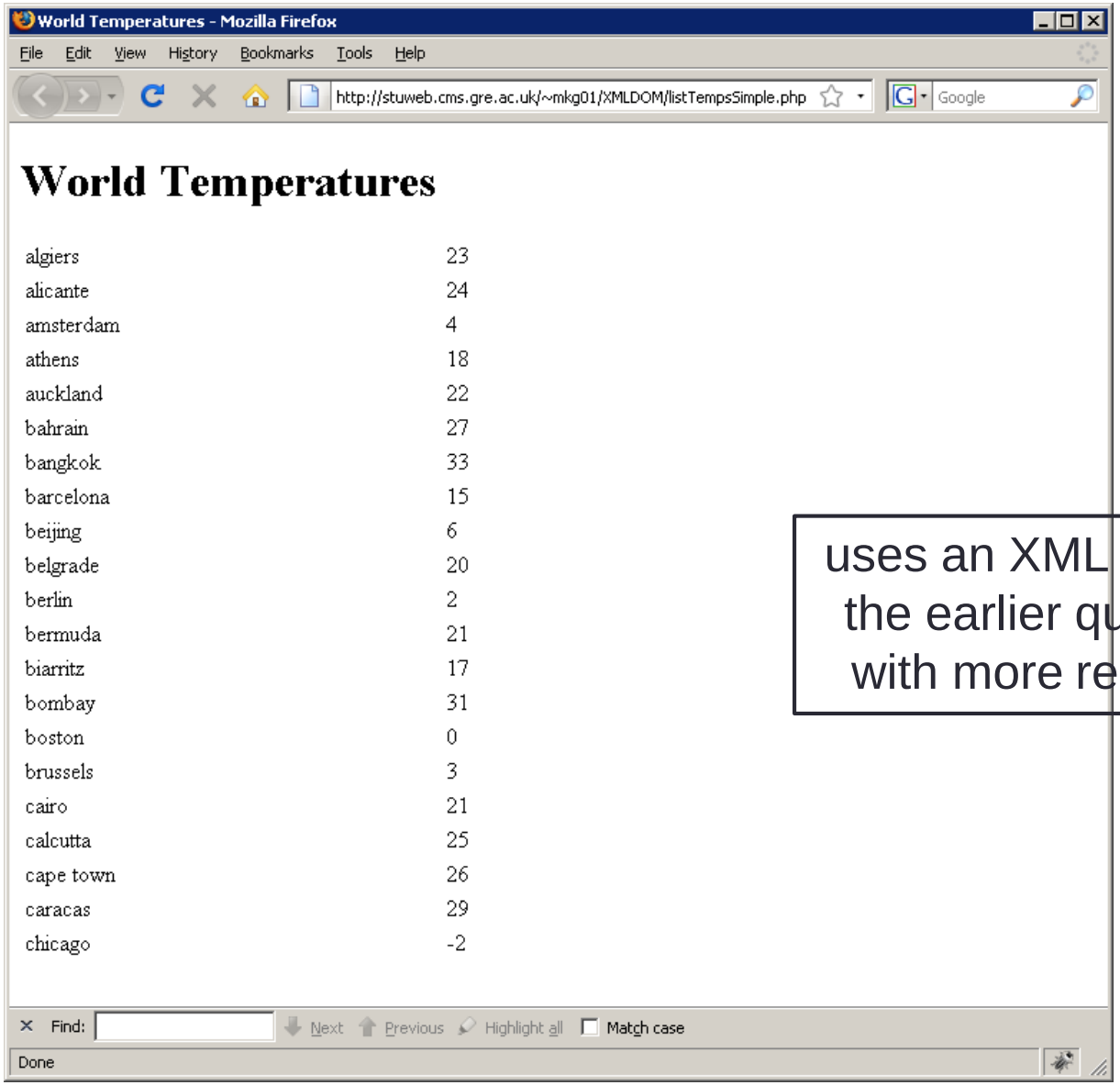
- Each object type described in the DOM has a standard set of methods and properties available.
- DOM language bindings (e.g. Java) provide a way of calling methods and accessing properties.
- e.g. Nodes (the base type for all entities in the document) includes the following properties:
  - **nodeName** is the name of the tag for nodes of type Element
  - **nodeValue** is the text value for a node of type Text

Node Relationships

The **childNodes** property of Element Starter would contain a **NodeList** of the two child nodes.



Server Side Processing With PHP



uses an XML file like the earlier quiz but with more records

listTempsSimple.php

```
<html><head><title>World Temperatures</title><head>
<body><h1>World Temperatures</h1>
<table width="50%">
<?php
$doc = new DOMDocument();

$xmlString = '';
foreach ( file('tempsDTD.xml') as $node ) {
        $xmlString .= trim($node);
}
$doc->loadXML($xmlString);

$records = $doc->documentElement->childNodes;
for ($i=0; $i<$records->length; $i++) {
        $townName = $records->item($i)->firstChild->textContent;
        $tempValue = $records->item($i)->lastChild->textContent;
        print "<tr><td>$townName</td><td>$tempValue</td></tr>\n";
}
?>
</table>
</body></html>
```

read the XML file into a single string with no whitespace

parse the XML string into a DOM structure in memory

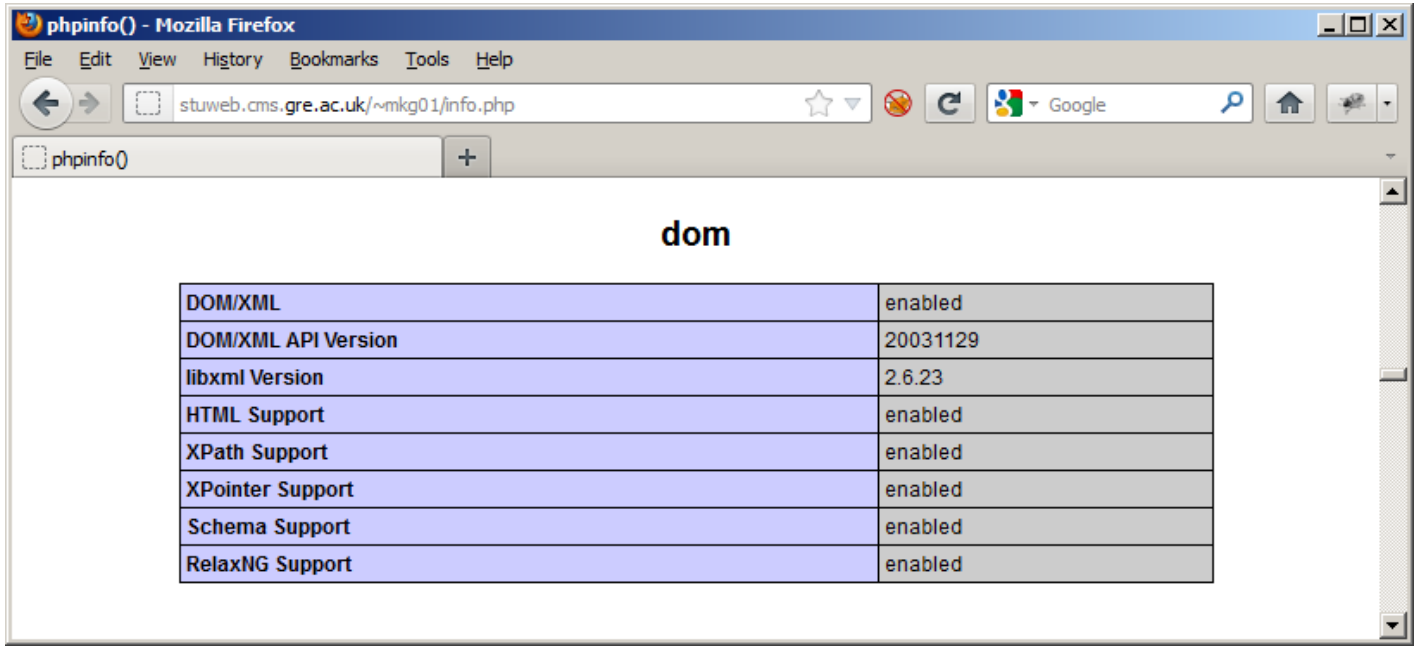
get the top node list **temperature_records**

loop over the child nodes

get the town names and temperature values

format the results as a table

Server Side Processing With PHP



PHP5 is bundled with a standard compliant DOM

PHP4 supported an ideosyncratic DOM

Reading and Parsing XML with PHP

- Reading and parsing the XML file requires that all whitespace is removed or you end up with twice as many nodes as you expected.

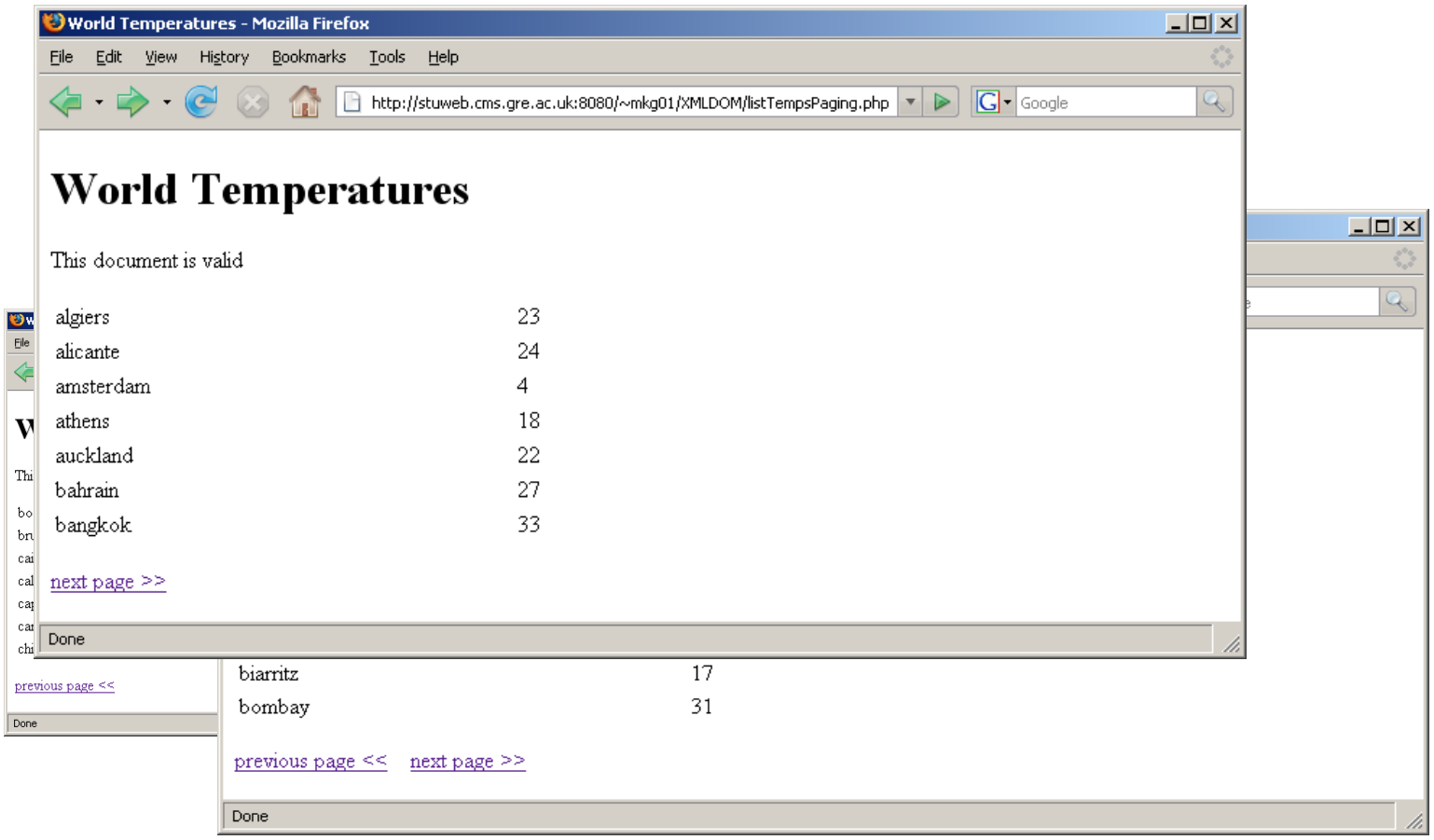
```
$xmlString = '';
foreach ( file('temperatures.xml') as $node ) {
    $xmlString .= trim($node);
}
$doc->loadXML($xmlString);
```

- There is a purpose built XML file reading and parsing function.

```
$doc->load('temperatures.xml')
```

  - but this requires removing whitespace from the XML file first
  - see listFlatTemps.php

listTempsPaging.php



listTempsPaging.php

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-gb">
<head>
<title>World Temperatures</title>
</head>
<body>
<h1>World Temperatures</h1>
<?php
$doc = new DOMDocument();
$xmlString = '';
foreach ( file('tempsDTD.xml') as $node ) {
          $xmlString .= trim($node);
}
$doc->loadXML($xmlString);

# Validate the XML against it's DTD
$valid = ( $doc->validate() ) ? 'valid' : 'not valid';
echo "<p>This document is $valid</p>\n";

$records = $doc->documentElement->childNodes;

# Calculate the first and last records
$pageLength = 7;
$first = ( isset($_GET['next']) ) ? $_GET['next'] : 0;
$last = ( $records->length - $first < $pageLength ) ?
          $records->length : $first + $pageLength;
?>
```

validate the XML against it's DTD

if it's the first call then "next" will not be set so start with the first record

calculate the last record on the page

listTempsPaging.php

```
<table width="50%">
<?php
# Print the records for this page
for ( $i=$first; $i<$last; $i++ ) {
          $townName = $records->item($i)->firstChild->textContent;
          $tempValue = $records->item($i)->lastChild->textContent;
          echo "<tr><td>$townName</td><td>$tempValue</td></tr>\n";
}
?>
</table>
<?php
# Format an anchor tag to link to the next page
echo '<p>';
if ( $last < $records->length - 1 ) {
   if ( $first > 0 ) {
      echo '<a href="listTempsPaging.php?next=' . ($last-2*$pageLength) .
           '">previous page &lt;&lt;</a>    ';
   }
   echo '<a href="listTempsPaging.php?next=' . $last .
           '">next page &gt;&gt;</a></p>';
} else {
   echo '<a href="listTempsPaging.php?next=' . ($last-2*$pageLength) .
            '">previous page &lt;&lt;</a></p>';
}
?>
</body></html>
```

output records from from first to last-1

- Can you do more with the DOM than this?
- Yes there are masses of properties and methods defined for reading and writing DOM objects
  - appendChild(), removeChild()
  - insertBefore(), insertAfter()
  - createElement()
- Refer to the W3C for the full specification

  http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/

  - although this is a less than easily readable document

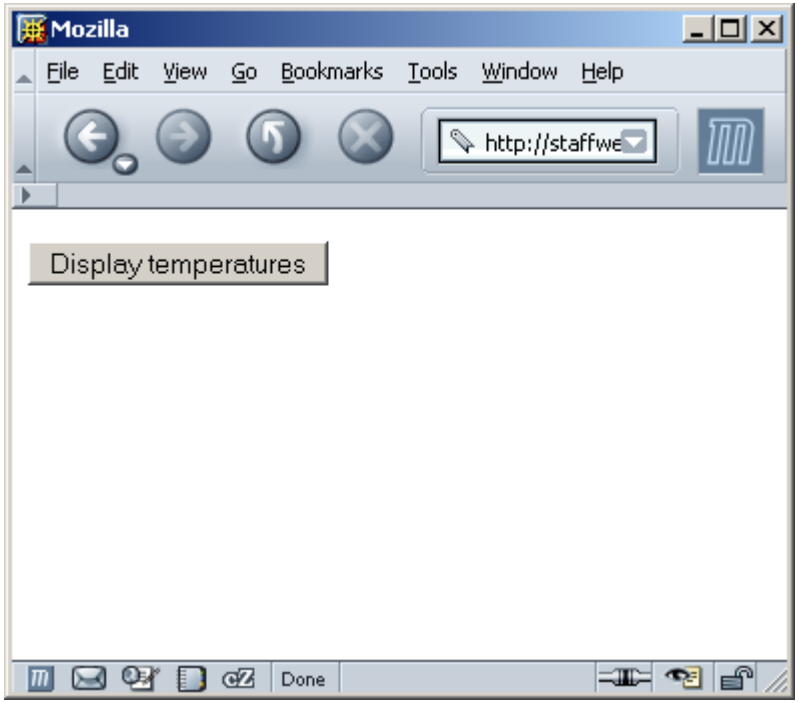
- Plenty of other resources

  http://www.mozilla.org/docs/dom/domref/dom_shortTOC.html

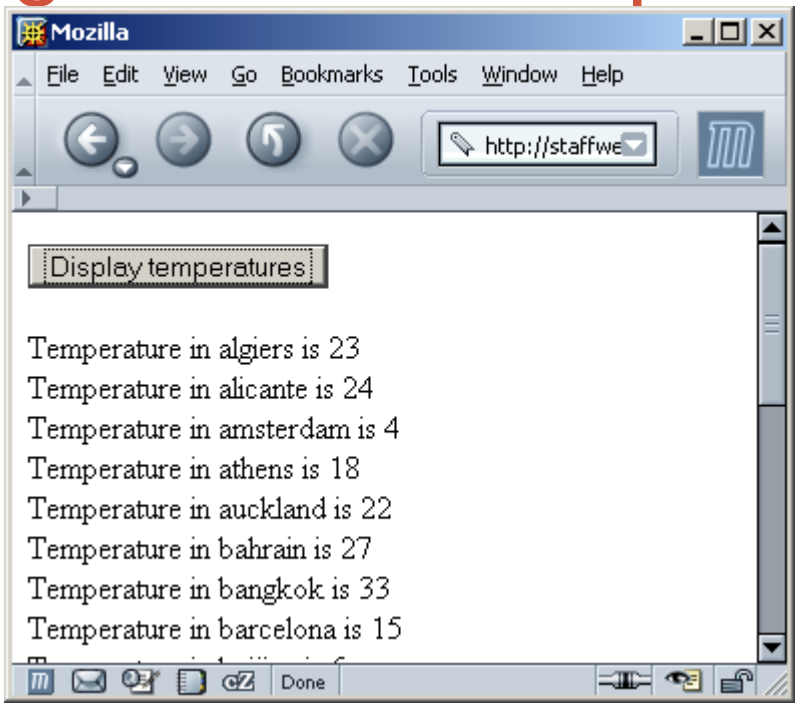
XML processing at the client-side

- The previous two examples show processing of an XML document on the server-side
  - XHTML was delivered to the client-side
  - no browser compatibility problems
- IE5+, NN6+ and Mozilla have considerable support for processing XML documents at the client-side
  - they allow XSLT and CSS style sheets to be applied to the document
- DOM on the client allows manipulation of a downloaded XML document using JavaScript
  - JavaScript also uses DOM to manipulate the XHTML document
- The following examples work on both IE and Netscape/Mozilla
  - not Opera
  - some interesting code to cope with browser variations

Client Side Processing With JavaScript



the XML is loaded into the browser with the HTML body onLoad event but is not visible



the button onClick event calls JavaScript code that loops through the DOM to display the records

listTempsStrict.html

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 //EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-gb">
<head>
<script type="text/javascript"><!--

// thanks to quirksmode.org for this idea
if ( document.implementation.createDocument ) {
   var doc = document.implementation.createDocument("", "", null);
} else if ( window.ActiveXObject ) {
   var doc = new ActiveXObject("Microsoft.XMLDOM");
} else {
   alert('Your browser can\'t handle this script');
}
```

Netscape and Mozilla

Internet Explorer

Instead of sniffing the browser by interrogating the navigator object this script tests the browser functionality in order to correctly instantiate an XML DOM object

listTempsStrict.html

```
function doIt() {
   var output = ""
   var records = doc.documentElement.childNodes;
   for ( var i = 0; i < records.length; i++ ) {
      townName = records[i].firstChild.firstChild.nodeValue;
      tempValue = records[i].lastChild.firstChild.nodeValue;
      output += "<br />Temperature in " + townName + " is " + tempValue;
   }
   document.getElementById("output").innerHTML = output;
}
--></script>

<title></title>
</head>
<body onload="doc.load('flatTemps.xml')">
<form action="dummy"><p>
<input type="button" onclick="doIt()" value="Display temperatures" />
<br /><span id="output"></span>
</p></form>
</body>
</html>
```

onLoad event handler loads the XML file into the XML DOM object

the output goes in here

listTempsStrict.html

- This example is much the same as the PHP version
  - JavaScript dot notation allows concatenation of node addressing

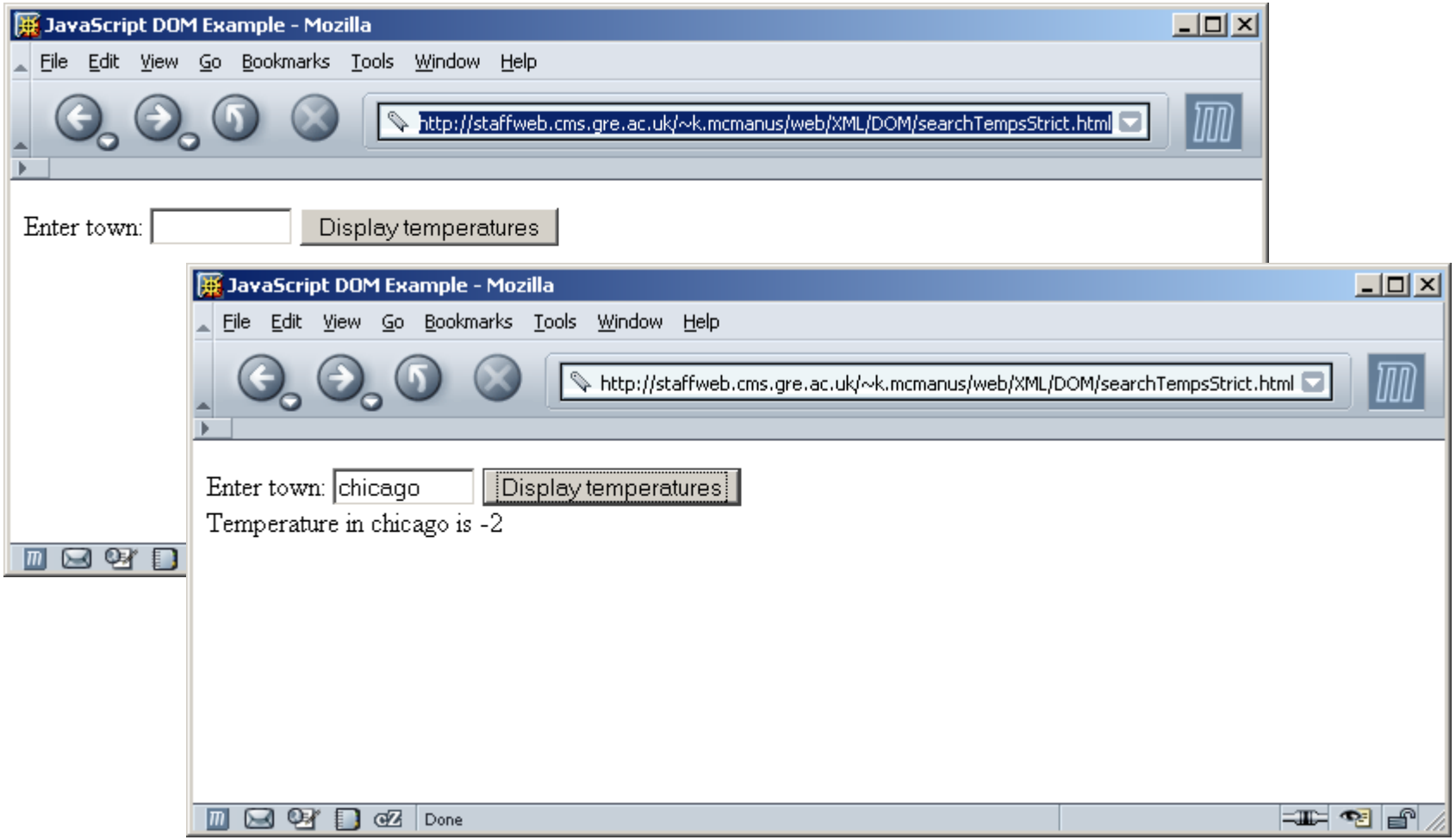
```
townName = records[i].firstChild.firstChild.nodeValue;
```

- Mozilla browsers offer the method **document.implementation.createDocument()**
  - accepts parameters for namespace and root node of the document should you need them
- Internet Explorer uses an **ActiveX object**
  - IE also supports **XML data islands**
  - a good idea but not part of XHTML 1.0 strict and not supported by other browsers

```
<xml id="XMLisland" src="booklist.xml">
```

- What does this example do that you couldn't do using an XSLT style sheet?

searchTempsStrict.html



searchTempsStrict.html

```
<snip>
function doIt() {
   var records = doc.documentElement.childNodes;
   found = false;
   for ( i= 0; i < records.length; i++ ) {
       townName = records[i].firstChild.firstChild.nodeValue;
       if (townName == document.forms[0].town.value) {
           tempValue = records[i].lastChild.firstChild.nodeValue;
           message = "Temperature in " + townName + " is " + tempValue;
           found = true;
       }
    }
    if (found == false) message = "Sorry not found";
    document.getElementById("output").innerHTML = message;
}
--></script>
</head>
<body onload="doc.load('flatTemps.xml')">
<form action="dummy"><p>
Enter town:
<input type="text" name="town" size="10"/>
<input type="button" onclick="doIt()" value="Display temperatures"/>
<br /><span id="output"></span>
</p></form>
</body></html>
```

loop over all records

if the record matches the form input data return the record

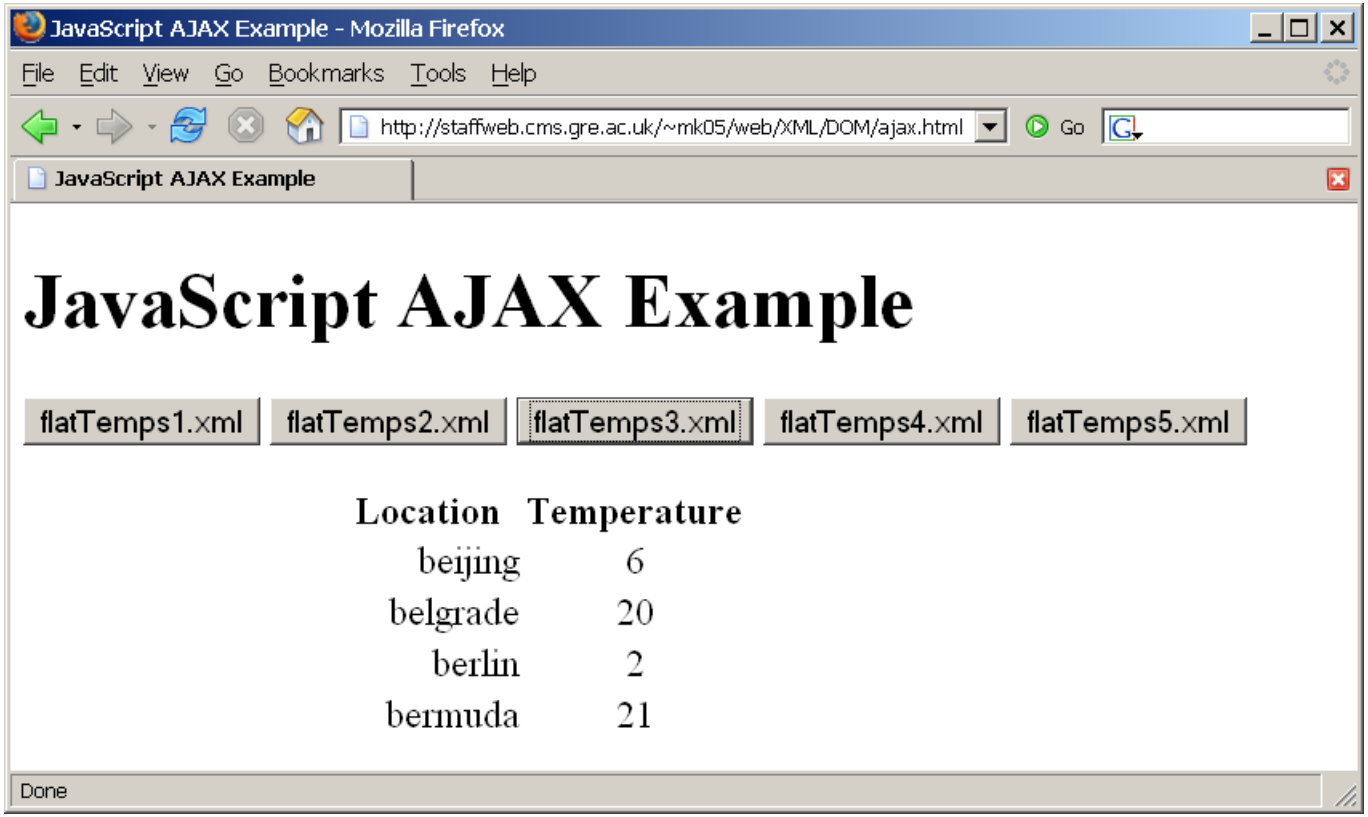
searchTempsStrict.html

- This is very similar to the previous example
- An XML DOM object is created by JavaScript in the HTML header
- An XML file is read into the XML DOM object when the page has been rendered
  - body tag onLoad event handler
  - this document is not displayed
- DOM is searched to match data from the input type text
  - better to implement this sort of application at the client rather than the server
    - why would this not be good with PHP?
- Not at all clear how you would do this with XSLT

AJAX

- **Asynchronous JavaScript and XML (AJAX)**
  - not a technology in itself
  - a "new" approach combining a number of existing technologies
    - XHTML
    - CSS
    - JavaScript
    - DOM
    - XML
    - XSLT
    - XMLHttpRequest object
- Web applications that make incremental updates
  - without reloading the entire browser page
  - faster and more responsive to user actions

ajax.html


JavaScript AJAX Example - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://staffweb.cms.gre.ac.uk/~mk05/web/XML/DOM/ajax.html
Go
JavaScript AJAX Example
JavaScript AJAX Example
flatTemps1.xml
flatTemps2.xml
flatTemps3.xml
flatTemps4.xml
flatTemps5.xml
Location Temperature
beijing 6
belgrade 20
berlin 2
bermuda 21
Done

ajax.html

```
if ( document.implementation.createDocument ) {
   var xmlDoc = document.implementation.createDocument("", "", null);
   xmlDoc.onload = doIt;
} else if ( window.ActiveXObject ) {
   var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
   xmlDoc.onreadystatechange = function () {
      if (xmlDoc.readyState == 4) doIt();
   }
} else {
   alert('Your browser can\'t handle this script');
}

function doIt() {
   var output = '<table><tr><th>   Location   </th><th>Temperature</th></tr>';
   var records = xmlDoc.documentElement.childNodes;
   for ( var i = 0; i < records.length; i++ ) {
      townName = records[i].firstChild.firstChild.nodeValue;
      tempValue = records[i].lastChild.firstChild.nodeValue;
      output += '<tr><td class="right">'+townName+'</td><td  class="centre">'+tempValue+'</td></tr>';
   }
   output += '</table>';
   document.getElementById("data").innerHTML = output;
}
--></script>
</head>
<body>
<h1>JavaScript AJAX Example</h1>
<p>
<input type="button" onclick="xmlDoc.load('flatTemps1.xml')" value="flatTemps1.xml" />
<input type="button" onclick="xmlDoc.load('flatTemps2.xml')" value="flatTemps2.xml" />
<input type="button" onclick="xmlDoc.load('flatTemps3.xml')" value="flatTemps3.xml" />
<input type="button" onclick="xmlDoc.load('flatTemps4.xml')" value="flatTemps4.xml" />
<input type="button" onclick="xmlDoc.load('flatTemps5.xml')" value="flatTemps5.xml" />
</p>
<div id="data"><table><tr><th>   Location   </th><th>Temperature</th></tr></table></div>
</body></html>
```

callback functions for the XML read completed event

event loads an XML file

ajax.html

- Page requests an XML document in response to a user event
  - button onClick
- The callback function **`doIt()`** is called when the XML document read completes
  - **`DoIt()`** parses the XML using DOM
  - updates the page content without re-loading the entire page
- AJAX applications usually use the XMLHttpRequest object
  - this one doesn't
  - for more information:
    - read the article by Jesse James Garrett
    - look at SAJAX – the Simple AJAX toolkit

XMLHttpRequest object

- Originally developed by Microsoft
  - Now widely supported
- Provides useful functionality
- Not necessarily asynchronous
  - use a callback mechanism like the previous example
- Not necessarily XML
  - txt, JSON, etc.

SAX – the Simple API for XML

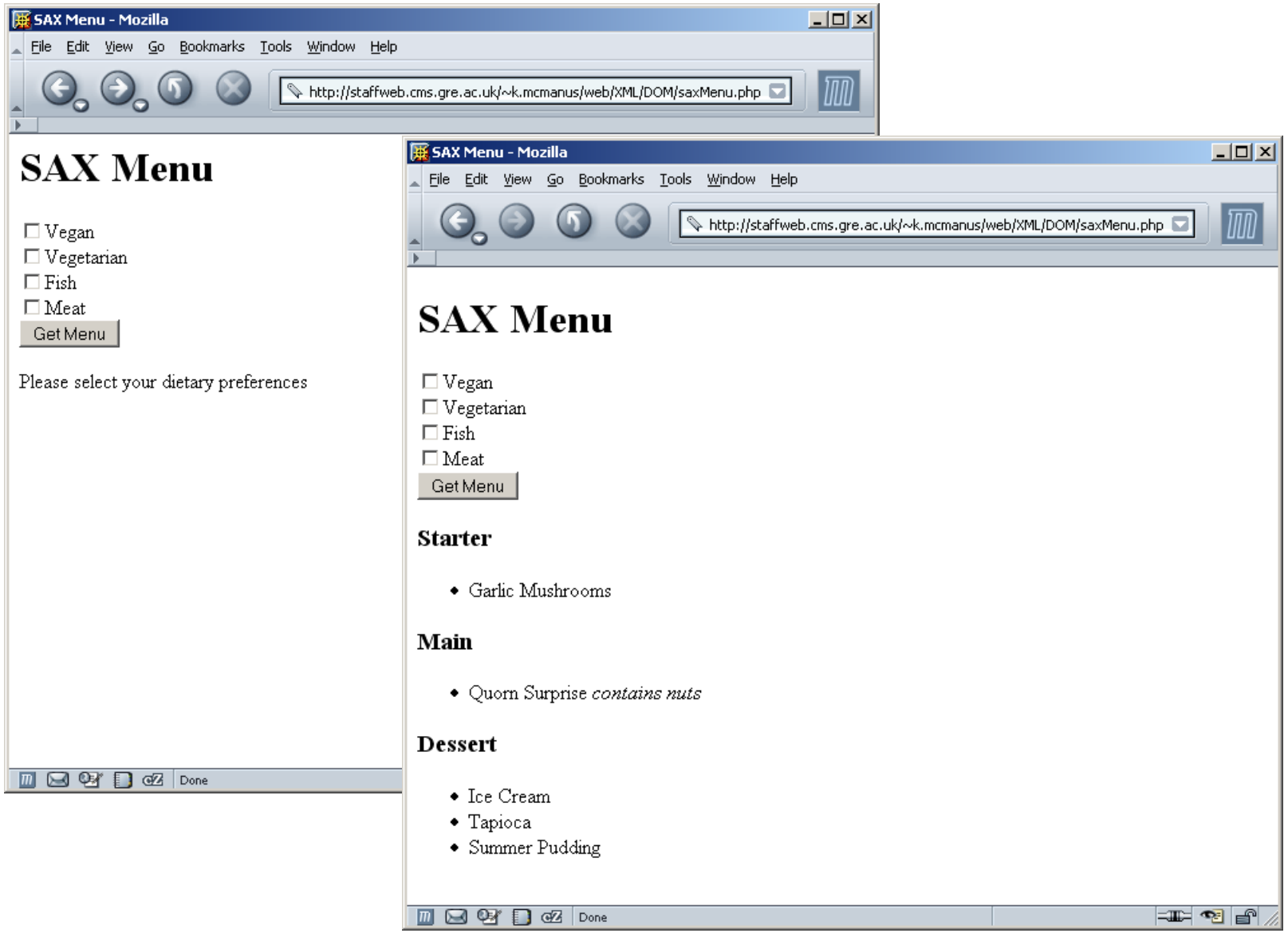
- An alternative to DOM
- SAX is a stream based processor
  - rather than reading the XML into memory the XML is processed as a stream
  - each XML tag encountered acts as an event that triggers a handler
- Current version has for some time been SAX 2.0.1
  - not a W3C standard – saxproject.org
- Originally Java only now available in several language bindings
  - PHP, Perl, Python, C++
- The “XML for <SCRIPT>” project provides a JavaScript DOM Level 2, Xpath and SAX processor for cross platform client side XML processing
- More efficient than DOM for many applications
  - SAX lacks the intuitive approach of DOM
  - less flexible
  - requires more code to achieve similar results
  - requires less memory to achieve similar results

SAX

“DOM and SAX have different philosophies on how to parse xml. The SAX engine is essentially event-driven. When it comes across a tag, it calls an appropriate function to handle it. This makes SAX very fast and efficient. However, it feels like you're trapped inside an eternal loop when writing code. You find yourself using many global variables and conditional statements.
On the other hand, the DOM method is somewhat memory intensive. It loads an entire XML document into memory as a hierarchy. The upside is that all of the data is available to the programmer. This approach is more intuitive, easier to use, and affords better readability.”

Matt Dunford

saxMenu.php


SAX Menu - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://staffweb.cms.gre.ac.uk/~k.mcmanus/web/XML/DOM/saxMenu.php
SAX Menu
Vegan
Vegetarian
Fish
Meat
Get Menu
Please select your dietary preferences
Done
SAX Menu - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://staffweb.cms.gre.ac.uk/~k.mcmanus/web/XML/DOM/saxMenu.php
SAX Menu
Vegan
Vegetarian
Fish
Meat
Get Menu
Starter
Garlic Mushrooms
Main
Quorn Surprise contains nuts
Dessert
Ice Cream
Tapioca
Summer Pudding
Done

saxMenu.php

```
<head><title>SAX Menu</title></head>
<body>
<h1>SAX Menu</h1>
<form action="saxMenu.php" method="post">
<p>
<input type="checkbox" name="chkVegan" />Vegan<br />
<input type="checkbox" name="chkVege" />Vegetarian<br />
<input type="checkbox" name="chkFish" />Fish<br />
<input type="checkbox" name="chkMeat" />Meat<br />
<input type="submit" name="getMenu" value="Get Menu" />
</p></form>

<?php
if ( isset( $_POST['getMenu']) ) {
    $diet = ( isset($_POST['chkVegan']) ? 'vegan ' : '' );
    $diet .= ( isset($_POST['chkVege']) ? 'vegetarian ' : '' );
    $diet .= ( isset($_POST['chkFish']) ? 'fish ' : '' );
    $diet .= ( isset($_POST['chkMeat']) ? 'meat ' : '' );
} else {
    die("<p>Please select your dietary preferences</p></body></html>\n");
}

// use the 'current' vars to keep track of which tag/attribute
// the parser is currently processing
$currentTag = '';
$currAttribs = '';
```

record the checkboxes in **$diet**

initialise some global variables to keep track of processing

saxMenu.php

```
// initialize parser
$xmlParser = xml_parser_create();
//disable case folding
xml_parser_set_option($xmlParser, XML_OPTION_CASE_FOLDING, false);

// set callback functions
xml_set_element_handler($xmlParser, 'startElement', 'endElement');
xml_set_character_data_handler($xmlParser, 'characterData');

// open XML file
$file = 'menu.xml';
if ( !($fp = fopen($file, 'r')) ) {
   die("<p>Cannot open XML data file: $file</p></body></html>\n");
}
// read and parse data
while ($data = fread($fp, 4096)) {
   if ( !xml_parse($xmlParser, $data, feof($fp)) ) {
      //error handling
      $errCode = xml_error_string(xml_get_error_code($xmlParser));
      $errLine = xml_get_current_line_number($xmlParser);
      xml_parser_free($xmlParser);
      die("<p>XML error: $errCode at line  $errLine</p></body></html>\n");
   }
}
xml_parser_free($xmlParser);
?>
</body></html>
```

read and parse the file in 4k blocks

liberate resources before exit

saxMenu.php

callback function for the opening tag

```
<?php
//callback functions
function startElement($parser, $name, $attribs) {
   global $currentTag, $currAttribs, $diet;
   $currentTag = $name;
   $currAttribs = $attribs;
   //define the HTML to use for the start tag
   switch ($name) {
   case 'Menu':
      break;
   case 'Dish':
      if ( strstr($diet,$currAttribs['diet']) ) {
         echo "<li>";
      }
      break;
   default:  // Starter, Main or Dessert
      echo '<h3>$name</h3><ul>';
      break;
   }
}
```

declare and update globals

if a “Menu” tag do nothing

if a “Dish” with a matching “diet” attribute start a list item

Otherwise it is a “Starter, “Main” or “Dessert” tag so start an unordered list

saxMenu.php

callback function for the closing tag

```
function endElement($parser, $name) {
   global $currentTag, $currAttribs, $diet;
   //output closing HTML tags
   switch ($name) {
   case 'Menu':
      break;
   case 'Dish':
      if ( strstr($diet,$currAttribs['diet']) ) {
         if ( isset($currAttribs['note']) ) {
            echo " <i>contains " . $currAttribs['note'] . '</i>';
         }
         echo '</li>';
      }
      break;
   default:  // Starter, Main or Dessert
      echo '</ul>';
      break;
   }
   //clear current tag variable
   $currentTag = '';
   $currAttribs = '';
}
```

declare globals

if a “Menu” tag do nothing

if a “Dish” with a matching “diet” attribute output any “note” and close the list item

Otherwise it is a “Starter, “Main” or “Dessert” tag so close the unordered list

saxMenu.php

callback function for the character data

```
//process data between tags
function characterData($parser, $data) {
   global $currentTag, $currAttribs, $diet;
   //add content to the HTML tags
   switch ($currentTag) {
   case 'Dish':
      if ( strstr($diet,$currAttribs['diet']) ) {
         echo $data;
      }
      break;
   default:
      echo $data; //includes the whitespace from the XML
      break;
   }
}
?>
```

declare globals

if a “Dish” with a matching “diet” attribute output the element content

view source in the browser to see this

saxMenu.php

- Three callback functions required to handle…
  - the opening tag
  - the closing tag
  - the element content
- Callback function parameters are defined in the specifications for **xml_set_element_handler()** and **xml_set_character_data_handler()**
  - hence the need for globals to keep track of processing
- The XML file is read as a stream triggering the callback functions
- Callback function conditionals determine the current action
  - tag name
  - attribute name
- Note how whitespace in the XML source affects the output
  - although not visibly in the browser
- Is this sensible use of SAX?
  - it is a single pass application
  - it could be run as a client side application

Summary

- DOM defines a standard, language independent, interface for manipulating XML documents.
- The view it presents to the programmer of the XML document is of a tree of objects.
- There are standard methods and properties available for the various object types
  - e.g. every node **parentNode** and a **firstChild** property
- Because the DOM allows you to manipulate an XML document using standard programming languages you have total flexibility in the processing you can perform
  - server and client side
- DOM is greedy
  - it creates a data structure in memory of the entire XML document
  - allows rapid navigation and modification of the document
- There are alternatives
  - do you really need heavyweight DOM when the lighter XSLT or SAX may be sufficient?

Questions?