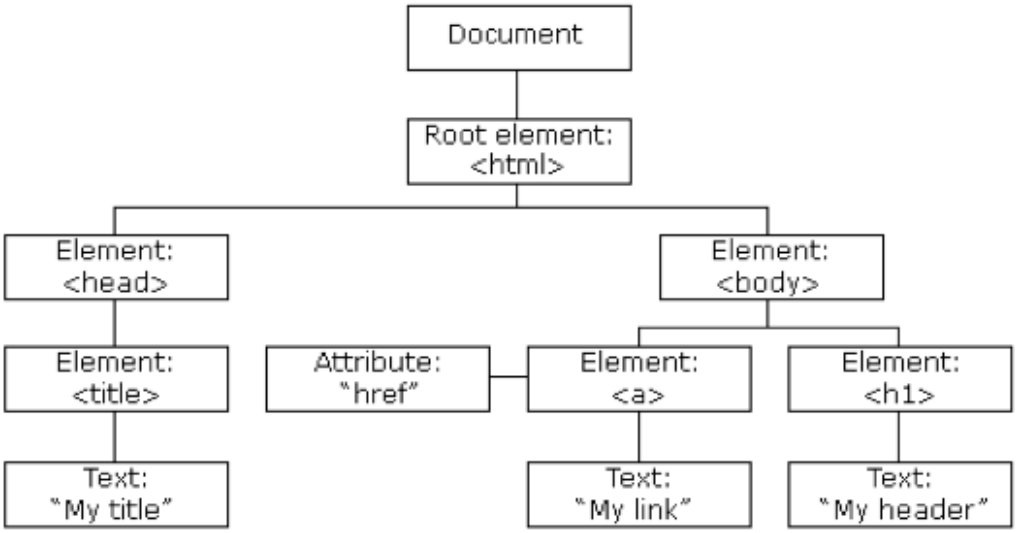
LECTURE 6: AJAX
Used to stand for ‘asynchronous javascript and xml’

Used for on-the-fly refreshing/updating (can refresh parts of a webpage rather than the whole thing by making use of the DOM:




- DOM - document object model - tree representation of XHTML/HTML
- Document
  - Root element
    - head
    - body
    - <h1>My header</h1>

- Normalized dom - one node (text node)
- attributes are linked list to the side of nodes
- Dom is a tree
  - in tree are different nodes (most common is element)
  - text nodes can be inside element nodes
  - attribute nodes can be “on the side”
  - comment, processing instruction nodes used less frequently

Javascript has methods that allow you to traverse and change doms
- AJAX is inserting content dynamically into a webpage

- Either insert raw HTML into page
- or insert new nodes into dom

- What makes AJAX possible?
  - Microsoft first made it possible in Internet Exploroer

- precursor to what is now supported by all major browsers
  - XMLHttpRequest - this object and methods and properties
  allows it to make other http requests without changing the whole page.
- References available (i.e., w3, Mozilla's representations)

METHODS
- have objects created by using new keyword
- open connection
- open(method (get, post), url (url), async (asynchronicity, browser not waiting for response while things are getting back to you - send content and immediately get back to user)
- send() - send http parameters
- setRequestHeader() used for caching reasons
- REAL MAGIC WITH open method and send data.
- synchronous means you have to wait in line--asynchronous means then, you don't

List of Methods:
Methods
abort()
getAllResponseHeaders()
getResponseHeader(header)
open(method, url)
open(method, url, async)
open(method, url, async, user)
open(method, url, async, user, password)
send()
send(data)
setRequestHeader(header, value)

<form onsubmit="quote(); return false;"> - return false means that submit actually isn't submitted
</form>

The “try – catch” statement:

- if try fails, do catch
- right way to do this is to have four try-catch blocks to try to catch one of four different versions

xhr.onreadystatechange = handler;
- telling xhr object, when receiving readystatechange state, invoke handler function

xhr.open("GET", url, true); - opens connection.. says true so that it is asynchronous
xhr.send(null); sends xhr

- as soon as response comes back from server, you get at the response via the event handler
  - changes from "i sent data" to "i received data"
- readyState values 4 (loaded)
- xhr.status=200 (means everything was ok)
- xhr.responseText - what a browser would have seen if page had loaded normally

- document.getElementById("price").value = xhr.responseText;
  - will put whatever came back as the value in the element with id price

AJAX3 - inserting xhr.responseText as a child of a span
  - document.getElementById("price").innerHTML = xhr.responseText;
    - clobbers whatever is within that span, and inserts the response text
    - removes bold element

different ways of inserting data directly into an HTML page
  - XHTML (text/HTML)
  - xml (text/HTML)
  - JSON (application/JSON)

what's bad about generating XHTML serverside?
- the time/cost of bringing it back

Generate XHTML text client-side!

Control visibility of XHTML element (display: none or block or inline/ visibility: on or off)
  - display: none;

Properties associated with xmlhttprequest object:

readyState
0 (unitialized)
1 (open)
2 (sent)
3 (receiving)
4 (loaded)

- onreadystatechange - something happens w
- readyState

  - 0 when you first created it
  - 4 when it gets back and is served
- responsetext - string returned
- responsebody - binary format
- responsexml - if what is returned is xml, and you want a dom to get returned, use this (parsed version of responsetext)
**Status: 200 is good, but anything else (404, 500, etc.) is bad)

When you get back response xml, you get actual xml code object (to be parsed client side)—and why do work serverside when you can do it clientside?

When you have a dom (even a mini-dom, like from responsexml) you can use dom functions
  - such as getElementsByTagName();

<price>444.44</price>  prices[0].firstChild.nodeValue;
  - firstChild is a text node, so get node value

node value vs. node name
 has node name "b" but no node value
444.44 has node value "444.44" but no node name

**You can use firebug to look inside as to what's going on inside browser and memory.

document.createElement("div"); returns reference to new node in memory who's tag is 
document.createTextNode() - creates a text node
div.appendChild(text); appends a text node to the div node
document.getelementbyid("quotes").appendchild(div) - puts div child in element with id quotes.

We can't use xpath clientside (yet!), only serverside—so, meet JSON:

JSON: javascript object notation

JSON, return an application/JSON
  - a string version of a javascript object
  - can the browser change this into a structure?

to evaluate JSON
var quote = eval("(" + xhr.responseText + ")");
  - converts string representation of an object into an object
  - what is stored in quote is a nameless object with three properties (price, high, and low)
  - to get price, just use quote.price

- JSON makes syntax easier

to output JSON (in quote6.PHP)
  print("{ price: $price, high: $high, low: $low }");
  and change mimetype to application/JSON

print(JSON_encode($stock));
  - creates JSON object for you, given a class called $stock (in quote7.PHP)

Frameworks:
Dojo
http://dojotoolkit.org/
Ext JS
http://extjs.com/
jQuery
http://jquery.com/
MooTools
http://mootools.net/
Prototype
http://www.prototypejs.org/
YUI
http://developer.yahoo.com/yui/
. . .