"Web 2.0"

- buzzword
  - probably originated with O'Reilly conference in 9/05
- what's different from "Web 1.0"?
  - [technical aspects, not business]
- Web as platform; cloud computing
  - systems as services, not on a PC, e.g., Google Docs
- continuous software update
  - because it's on the server
- data as central component
  - e.g., Amazon, Google, Yahoo, Facebook, Flickr, ...
- lightweight programming & data transfer
  - Atom, REST instead of SOAP
  - JSON instead of XML
- mashups using APIs
  - Google maps, Yahoo pipes,
- "collective intelligence" (?)
  - Wikipedia, Google page rank, online reviews, blogs, crowd-sourcing, Twitter, ...

XMLHttpRequest ("XHR")

- interactions between client and server are usually synchronous
  - there can be significant delay
  - page has to be completely redrawn
- XMLHttpRequest provides asynchronous communication with server
  - often no visible delay
  - page does not have to be completely redrawn
- first widespread use in Google Suggest, Maps, Gmail (Feb 2005)
  - "The real importance of Google’s map and satellite program, however, is not its impressive exterior but the novel technology, known as Ajax, that lies beneath." (James Fallows, NY Times, 4/17/05)
- Ajax: Asynchronous Javascript + XML
  (shorthand/marketing/buzzword term coined 2/05)
  - (X)HTML + CSS for presentation
  - DOM for changing display
  - Javascript to implement client actions
  - XML for data exchange with server (but it doesn’t have to use XML)
  - "server agnostic": server can use any technology
Ajax interface to Princeton directory

<h1> unPhonebook</h1>

<form name=phone>
Type here:
<input type="text" id="pat"
    onkeyup='geturl(pat.value);' >
</form>

<pre id="place"></pre>

unPhonebook

Start typing here: kernighen

Brian W Kernighan (bwk)  609-258-2089  311 Computer Science  Computer Science

Basic structure of Ajax code

var req;

function geturl(s) {
    if (s.length > 1) {
        url = 'http://www.cs.princeton.edu/~bwk/phone3.cgi?' + s;
        loadXMLDoc(url); // loads asynchronously
    }
}

function loadXMLDoc(url) {
    if (window.XMLHttpRequest) { // native XMLHttpRequest
        req = new XMLHttpRequest();
    } else if (window.ActiveXObject) { // IE ActiveXObject
        req = new ActiveXObject("Microsoft.XMLHTTP");
    }
    if (req) {
        req.onreadystatechange = processReqChange;
        req.open("GET", url);
        req.send(null);
    }
}

function processReqChange() {
    if (req.readyState == 4) { // completed request
        if (req.status == 200)
            show(req.responseText); // or responseXML
    }
}

function show(s) { // show whatever came back
    document.getElementById("place").innerHTML = s
}
Server script (phone2.cgi)

```
q1=`echo $QUERY_STRING | gawk '{split($0,x,"%20"); print x[1]}'
q2=`echo $QUERY_STRING | gawk '{split($0,x,"%20"); print x[2]}'
/usr/local/bin/ldapsearch -x -h ldap.princeton.edu -u -b 
-o='Princeton University,c=US' "(cn=*$q1*)" uid cn telephoneNumber 
studenttelephoneNumber studentstreet street ou |
php -r 'while (!feof(STDIN)) {
  $d = (fgets(STDIN));
  if (preg_match("/\#/", $d)) continue;
  if (preg_match("/^dn:|^ufn:/", $d)) continue;
  if (preg_match("/^cn:/", $d))
    if (strlen($d) > strlen($cn)) $cn = $d;
  if (preg_match("/telephoneNumber|street/", $d))
    $out = $out . " " . trim($d);
  if (preg_match("/^ou:/", $d)) $out = $out . " " . trim($d);
  if (strlen(trim($d))==0 && strlen($cn . $out) > 0) {
    $out = trim($cn) . " " . $out;
    $out = preg_replace("/Undergraduate Class of/", "", $out);
    $out = preg_replace("/cn:|ou:|telephoneNumber:|(student)?street:/" 
    $out = preg_replace("/@Princeton.EDU/", "", $out);
    print "$out\n";
    $out = $cn = "";
  }
}' | grep -i ".*$q2" | sed -e /Success/d
```

Simpler server script (phone3.cgi)

```
#!/bin/sh
PATH=.:/usr/bin:/usr/local/bin

echo "Content-Type: text/html"; echo

q1=`echo $QUERY_STRING | 
gawk '{ n=split($0, x, "%20"); print x[1]}'
q2=`echo $QUERY_STRING | 
gawk '{ n=split($0, x, "%20"); print x[2]}'
q3=`echo $QUERY_STRING | 
gawk '{ n=split($0, x, "%20"); print x[3]}'

grep -i "$q1" phone.txt |
grep -i ".$q2" |
grep -i ".$q3"

• works on precomputed data file
```
Javascript objects

• everything in Javascript is an object
  - except numbers, booleans, null, undefined

• create objects with
  var obj = new Object();
  var obj = {};

• objects are collections of named values
  - name-value pairs
  - essentially just associative arrays
  - can access elements with either syntax
    obj.property = whatever;
    obj["property"] = whatever;

• values can be anything
  - basic values like numbers
  - arrays
  - functions
  - objects

Javascript objects (2)

• function literals
  - functions are just values
    var max = function(a,b) { if (a>b) return a; else return b; }

• object literals (initializers):

  var course = {
    dept: "cos",
    numbers: [109, 333],
    prof: {
      name1: "brian", name2: "kernighan",
      office: { bldg: "cs", room: "311" },
      email: "bwk"
    },
    toString: function() {
      s = this.dept + this.numbers + " "
      + this.prof.name1 + " " + this.prof.name2 + " "
      + this.prof.office.bldg + this.prof.office.room
      + " " + this.prof.email;
      return s
    }
  }
Javascript objects (3)

- each object has a prototype property that is used to make new instances
- changing the prototype affects all subsequent ones

```javascript
function Point(x,y) {
    this.x = x; this.y = y;
}
Point.prototype.dist = function(that) {
    var dx = this.x - that.x;
    var dy = this.y - that.y;
    return Math.sqrt(dx*dx+dy*dy);
}
Point.prototype.toString = function() {
    return '(' + this.x + ',' + this.y + ')';
}
Point.ORIGIN = new Point(0,0);
var p = new Point(3,4);
var d = p.dist(Point.ORIGIN);
var msg = "Dist to " + p + " is " + d;
```

JSON : Javascript Object Notation

- lightweight data interchange format
  - based on object literals
  - an alternative to XML
  - maps easily to most other languages
- two basic structures
  - object: unordered collection of name-value pairs
    just an associative array or hash table
    `{ string: value, string, value, ... }`
  - array: ordered collection of values
    `[ value, value, ... ]`
  - string is ""...
  - value is string, number, true, false, object or array
- Javascript eval function can convert this into a data structure:
  ```javascript
  var obj = eval(json_string) # bad idea!
  - this is potentially unsafe, since the string can contain more than just JSON
  ```
- see json.org
YAML

*YAML 1.2
---
YAML: YAML Ain’t Markup Language

What It Is: YAML is a human friendly data serialization standard for all programming languages.

YAML Resources:
- YAML 1.0 (1st Edition): [http://yaml.org/spec/1.0/](http://yaml.org/spec/1.0/)
- YAML Trac Wiki: [http://trac.yaml.org/](http://trac.yaml.org/)
- YAML Mailing List: [yaml-core@lists.sourceforge.net](mailto:yaml-core@lists.sourceforge.net)
- YAML IRC Channel: [#yaml on irc.freenode.net](irc.freenode.net)
- YAML Cookbook (Ruby): [http://yaml4r.sourceforge.net/cookbook/](http://yaml4r.sourceforge.net/cookbook/)

Projects:
- **C/C++ Libraries:**
  - *libyaml* # "C" Fast YAML 1.1
  - *PyYaml* # (dated) "C" YAML 1.0
  - *yaml-cpp* # C++ YAML 1.1 implementation
- **Java:**
  - *JvYaml* # Java port of RbYaml
  - *SnakeYAML* # Java 5 / YAML 1.1
  - *YamlBeans* # To/from JavaBeans
  - *JYaml* # Original Java Implementation

Libraries, API’s, Frameworks

- browsers are not perfectly standardized
- DOM and CSS coding is messy and complicated
- web services are ever more complex

- how do we make it easy to create applications?

- libraries of common Javascript operations

- API’s, often Javascript, to access services

- frameworks: development environments for integrated client & server programming
From developer.yahoo.com

YAHOO.util.Connect = {
    _msxml_progid:
        ['MSXML2.XMLHTTP.5.0',
         'MSXML2.XMLHTTP.4.0',
         'MSXML2.XMLHTTP.3.0',
         'MSXML2.XMLHTTP',
         'Microsoft.XMLHTTP'],
    createXhrObject:function(transactionId) {
        var obj,http;
        try {
            http = new XMLHttpRequest();
            obj = { conn:http, tId:transactionId };
        }
        catch(e) {
            for (var i=0; i<this._msxml_progid.length; ++i){
                try {
                    http = new ActiveXObject(this._msxml_progid[i]);
                    obj = { conn:http, tId:transactionId };
                    break;
                } catch(e){}
            }
        }
        finally {
            return obj;
        }
    }, ...

Javascript libraries

- library of Javascript functions that typically provides
  - easier access to DOM
  - convenience functions for arrays, iterators, etc.
  - uniform interface to Ajax
  - visual effects like fading, flying, folding, ...
  - drag and drop
  - in-place editing
  - extensive set of widgets: calendar, slider, progress bar, tabs, ...

- there are lots of these!
  - Prototype & Scriptaculous, Dojo, jQuery, MochiKit, MooTools,
    Yahoo User Interface (YUI) ...

- see code.google.com/apis/ajaxlibs/
  - single library for uniform access to ~10 Javascript libraries
Callbacks

- callback: a function that is passed as an argument to another function, and executed after the parent function has been executed
  - functions can be passed around like variables
- callback with no argument
  
  ```javascript
  foo(args, myCallback);
  ```
- callback with arguments: anonymous function that calls the callback when invoked
  
  ```javascript
  foo(args, function() {
      myCallback(param1, param2);
  });
  ```
  - still have to get the arguments to it
- Google maps uses this a lot
  
  ```javascript
  getLatLng(address:String, callback:function)
  ```
  - "Sends a request to Google servers to geocode the specified address. If the address was successfully located, the user-specified callback function is invoked with a GLatLng point. Otherwise, the callback function is given a null point."