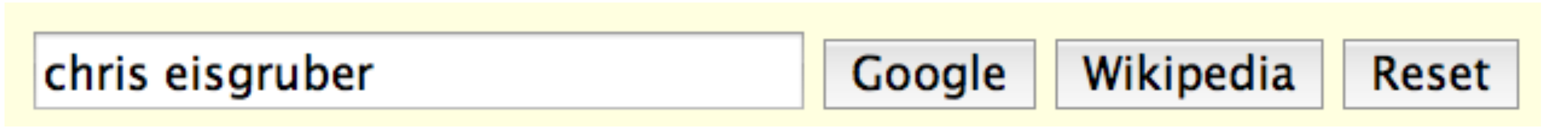
DOM: Document Object Model

- **browser presents an object interface**
  - accessible from and modifiable by Javascript
- **DOM entities have methods, properties, events**
  - element properties can be accessed & changed
  - elements can be added or removed
- **document object holds page contents**
  - elements stored in a tree: HTML tags, attributes, text, code, ...
  - each element is accessible through the DOM
  - through functions called from Javascript
- **page is "reflowed" (smart redraw) when anything changes**

- **window object also has methods, properties, events**
  - alert(msg), prompt(msg), open(url), ...
  - size, position, history, status bar, ...
  - onload, onunload, ...
  - window.document: the document displayed

Basic events on forms



```
<head>
<script>
function setfocus() { document.srch.q.focus(); }
</script>
</head>
<body onload='setfocus();'>
<H1>Basic events on forms</H1>
<form name=srch
      action="http://www.google.com/search?q="+srch.q.value>
<input type=text size=25
       id=q name=q value="" onmouseover='setfocus()'>
<input type=button value="Google" name=but
    onclick='window.location="http://www.google.com/
                                 search?q="+srch.q.value'>
<input type=button value="Wikipedia" name=but
    onclick='window.location="http://en.wikipedia.com/
                                     wiki/"+srch.q.value'>
<input type=reset onclick='srch.q.value=""' >
</form>
```

More examples...

- **in a form:**

```
<form>
 <input type=button value="Hit me"
    onClick='alert("Ouch! That hurt.")'> <P>
 <input type=text name=url size=40 value="http://">
 <input type=button value="open"
    onClick='window.open(url.value)'> <P>
 <input type=text name=url2 size=40 value="http://">
 <input type=button value="load"
    onClick='window.location=url2.value'> <P>
 <input type=button value="color it "
    onClick='document.bgColor=color.value'>
 <input type=text name=color value='type a color'>
 <input type=button value='make it white'
    onClick='document.bgColor="white"'>
</form>
```

- **in a tag**

```
<body onLoad='alert("Welcome to my page")'>
```

- **on an image**

```
<img src="smiley.jpg" onMouseover='src="frowny.gif"'
                      onMouseout='src="smiley.jpg"'>
```

- **etc.**

Dynamic CSS

- **style properties can be set dynamically**
  - color, alignment, border, margins, padding, ...
  - for individual elements, or all elements of a type, or of a given name
  - can be queried and set by Javascript

```
<script>
  window.onload = function() {
     var p = document.getElementsByTagName("P");
     for (var i = 0; i < p.length; i++) {
        p[i].onmouseover = function() {
           this.style.backgroundColor = "#deadbe";
        };
        p[i].onmouseout = function() {
           this.style.backgroundColor = "white";
        };
     }
  }
</script>
```

CSS dynamic positioning

- **DOM elements have "style" attributes for positioning**
  - a separate component of CSS
  - provides direct control of where elements are placed on page
  - elements can overlap other elements
    - on separate layers
- **basis of animation, drag & drop**
- **often controlled by Javascript**

```
<img src="dog.jpg" id="dog" onClick='hit()'
    style="position:absolute; left:100px; top:60px" >
```

```
var dog = document.getElementById("dog")
dog.style.left = 300 * Math.random() + "px"
dog.style.top = 300 * Math.random() + "px"
```

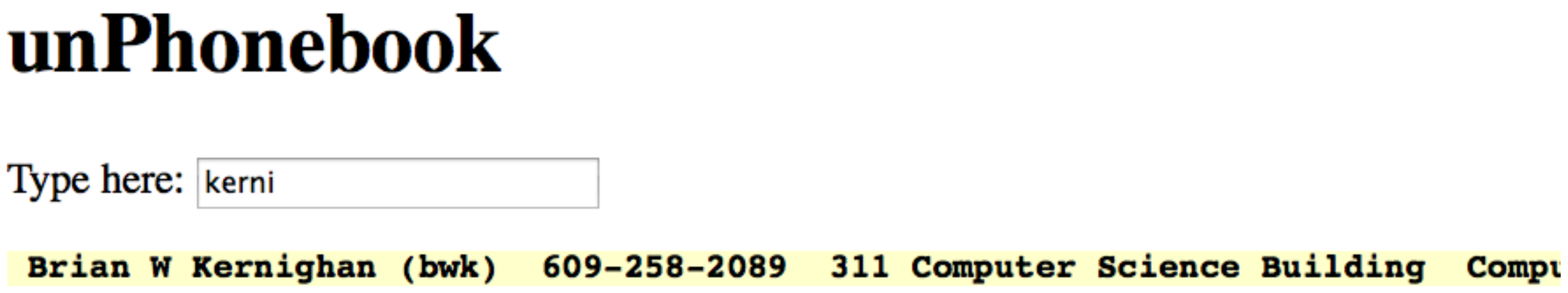
XMLHttpRequest ("XHR")

- **interactions between client and server are usually synchronous**
  - there can be significant delay
  - page has to be completely redrawn
- **XMLHttpRequest provides <u>asynchronous</u> 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 and 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

Type here: kerni

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

Basic structure of Ajax code in browser

```
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) {
   req = new XMLHttpRequest();
   if (req) {
      req.onreadystatechange = processReqChange;
      req.open("GET", url);
      req.send(null);
   }
}
function processReqChange() {
   if (req.readyState == 4) {    // completed request
      if (req.status == 200)     // successful
         show(req.responseText); // could be responseXML
   }
}
function show(s) {   // show whatever came back
   document.getElementById("place").innerHTML = s
}
```

XHR with nested function definition

```
function loadXMLDoc(url) {
   req = new XMLHttpRequest();
   if (req) {
      req.onreadystatechange = function() {
         window.status = req.statusText;
         if (req.readyState == 4) {  // completed request
            if (req.status == 200)   // successful
               show(req.responseText);
         }
      };
      req.open("GET", url);
      req.send(null);
   }
}
```

Callbacks

- **callback: a function that is passed as an argument to another function, and executed sometime later**
  - functions can be passed around like variables
    - e.g., function pointers in C; like ordinary variables in most languages
- **extensively used in Javascript because we don't want the browser to block waiting for response**

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

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**

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**
  - especially access to DOM

- **packages for layout with CSS**

- **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

- **Javascript functions that typically provide some combination of**
  - 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
  - widget set4: calendar, slider, progress bar, tabs, ...

- **there are lots of such libraries**
  - jQuery, jQueryUI, React.js, Angular.js, Backbone.js ...

Basic structure of Ajax code in browser

```
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) {
   req = new XMLHttpRequest();
   if (req) {
      req.onreadystatechange = processReqChange;
      req.open("GET", url);
      req.send(null);
   }
}
function processReqChange() {
   if (req.readyState == 4) {    // completed request
      if (req.status == 200)     // successful
         show(req.responseText); // could be responseXML
   }
}
function show(s) {   // show whatever came back
   document.getElementById("place").innerHTML = s
}
```

jQuery example

```
<script>
   function geturl(s) {
      if (s.length > 1) {
         var url = 'http://www.cs.princeton.edu/
                                        ~bwk/phone3.cgi?' + s;
         $.get(url, function(res) {
            $('pre').empty().append(res);;
         });
      }
   }
</script>
<form name=phone>
Type here:
<input type="text" id="pat" onkeyup='geturl(pat.value);'>
</form>
<pre id="place"></pre>
```

Debugging Javascript

- **it's hard**
- **use var declarations, check balanced quotes, braces, brackets, …**
- **use the debugger**

- **in Chrome**
  - "tools menu" / More tools / Javascript console
- **in Firefox**
  - Tools / Web developer / Web Console
- **in Safari**
  - Develop / Web console

- **use console.log to write debugging output**
  - like printf
  - much better than alert(…) for most things