DOM: Document Object Model

- **browser presents an object interface**
  - accessible from Javascript
- **window object has methods, properties, events**
  - alert(msg), prompt(msg), open(url), ...
  - size, position, history, status bar, ...
  - onload, onunload, ...
  - window.document: the document displayed
- **document object holds page or frame contents**
  - elements stored in a tree
    tags, attributes, text, ...
  - each element is accessible through the DOM
  - through functions called from Javascript

- **element properties can be accessed & changed**
- **elements can be added or removed**

- **page is "reflowed" (smart redraw) when anything changes**

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 onUnload='alert("bugging out")'>
```

- **on an image**

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

- **etc.**

CSS: Cascading Style Sheets

- **a language describing how to display (X)HTML documents**
- **separates structure (HTML) from presentation (CSS)**
- **style properties can be set by declarations**
  - for individual elements, or all elements of a type, or with a particular name
- **can control color, alignment, border, margins, padding, …**

```
<style type="text/css" media="all">
    body { background: #fff; color: #000; }
    pre { font-weight: bold; background-color: #ffffcc; }
    a:hover { color: #00f; font-weight: bold;
              background-color: yellow; }
</style>
```

- **style properties can be queried and set by Javascript**

```
<body id="body">
<script>
    var b = document.getElementById("body")
    b.style.backgroundColor='lightyellow'
    b.style.fontFamily='Verdana'; b.style.fontSize='14px'
    b.style.color='blue'
</script>
```

CSS syntax

- **optional-selector { property : value; property : value; … }**
- **selectors:**
  - HTML tags like h1, p, div, …
  - .classname (all elements with that classname)
  - #idname (all elements with that idname)
  - :pseudo-class (all elements of that pseudo-class, like hover)

```
h1 { text-align: center; font-weight: bold; color: #00f }
h2, h3 { margin:0 0 14px; padding-bottom:5px; color:#666; }
.big { font-size: 200%; }
```

- **styles can be defined inline or (better) from a file:**

```
<link rel="stylesheet" href="mystyle.css">
```

- **can be defined in `<style> ... </style>` tag**
- **can be set in a `style="..."` attribute in an element tag**

```
<p class=big style="color:red">
```

More CSS examples (with thanks to notes by Austin Walker '13)

- select all elements

```
div { text-align: left; }
```

- select all elements with id princeton

```
#princeton { color:#FF9933; font-size: 200%; }
```

- select all elements with class yale

```
.yale { color:#AA0000; font-size: 50%; }
```

- select all <P> elements with class harvard

```
p.harvard { color: 0000AA; font-size: 10%; }
```

Dynamic CSS

- **style properties can be set dynamically**
  - color, alignment, border, margins, padding, ...
  - for individual elements, or all elements of a type
  - 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 = "yellow";
        };
        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 for 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"
```

Other HTML stuff

- **specialized markups**
  - SVG (scalable vector graphics)
  - Canvas Tags (scriptable graphics)
  - HTML5 (next version, to help replace Flash, Silverlight, etc.)

- **XUL (XML user interface language)**
  - built from CSS, Javascript, DOM
  - analogous extension mechanisms in Chrome and Safari
  - portable definition of common widgets like buttons

- **browser plug-ins and extensions**
  - Firebug
  - Greasemonkey

- ...

XMLHttpRequest ("XHR")

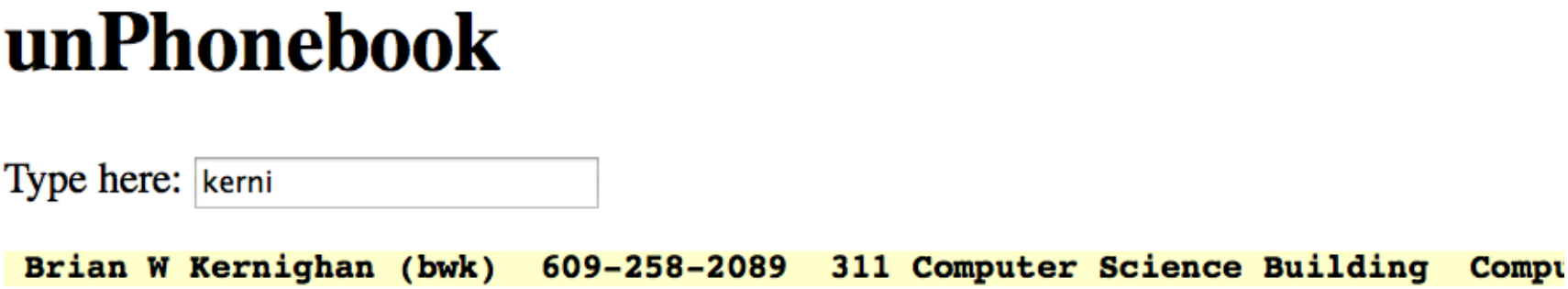
- **interactions between client and server are usually synchronous**
  - there can be significant delay
  - page has to be completely redrawn
- **XMLHttpRequest provides <u>asynchronous</u> communication with server**
  - often no visible delay
  - page does not have to be completely redrawn
- **first 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 + 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>
```



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
}
```

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**

```
foo(args, myCallback);
```

- **callback with arguments: anonymous function that calls the callback when invoked**

```
foo(args, function() {
  myCallback(param1, param2);
});
```

  - still have to get the arguments to it

XHR with callback

```
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);
    }
}
```

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**