Dynamic web interfaces

- **forms are a limited interface**

```
<FORM METHOD=GET
  ACTION="http://campuscgi.princeton.edu/~bwk/hello1.cgi" >
<INPUT TYPE="submit" value="hello" >
</FORM>
```

- **limited interaction on client side**
  - e.g., Javascript for simple validation
- **form data sent to server for processing**
- **synchronous exchange with server**
  - potentially slow: client blocks waiting for response
- **recreates entire page with what comes back**
  - even if it's mostly identical to current content

- **how can we make web interfaces more interactive and responsive?**
- **dynamic HTML: HTML + CSS, DOM, Javascript**
- **asynchronous partial update: XMLHttpRequest / Ajax**
- **plugins like Flash, Silverlight, ...**

Javascript

- **client-side scripting language (by Brendan Eich at Netscape, 1995)**
  - C/Java-like syntax
  - weakly typed, basic data types: double, bool, string, array, object
  - object-oriented, very dynamic
    unusual object model based on prototypes, not classes
- **usage:**

```
<script> javascript code </script>
<sometag onSomeEvent ='javascript code'>
<script src="url "></script>
```

- **can catch events from mouse, keyboard, ...**
- **can access browser's object interface**
  - window object for window itself
  - document object (DOM == document object model) for entities on page
- **can change a page without completely redrawing it**

- **lots of incompatibilities among browsers**
  - HTML, DOM, Javascript all potentially vary

Find the largest number

```
<html>
<body>
<script>

 var max = 0
 var num
 num = prompt("Enter new value")
 while (num != null && num != "") {
    if (parseFloat(num) > max)
       max = num
    num = prompt("Enter new value")
 }
 alert("Max = " + max)

</script>
</body>
</html>
```

- needs parseInt or parseFloat to coerce string value to a number

ATM checksum

```
function atm(s) {
   var n = s.length, odd = 1, sum = 0
   for (i = n-1; i >= 0; i--) {
      if (odd)
         v = parseInt(s.charAt(i))
      else
         v = 2 * parseInt(s.charAt(i))
      if (v > 9)
         v -= 9
      sum += v
      odd = 1 - odd
   }
   if (sum % 10 == 0)
      alert("OK")
   else
      alert("Bad.  Remainder = " + (sum % 10))
}

<form name=F0 onsubmit="">
 <input type=text name=num >
 <input type=button value="ATM"
    onClick='atm(document.forms.F0.num.value)'>
 <input type=reset value="Reset">
</FORM>
```

Javascript on a page

- **case sensitive**
- **semicolons or newline as statement terminators**
- **// or /*...*/ comments**
- **var x to declare variable**
  - scope is either global or local to current function
- **double, bool, 'string' or "string" with \ escapes**
  - null for undefined value
- **operators, expressions, control flow like C, Java**

```
if-else, while, for, do-while, switch, ...
for (v in obj) ...
try {...} catch( ) {...} finally {...}
```

- **user-defined functions**

```
function sum(x, y) { return x + y; }
```

- **arrays are objects**

```
var a = [zero, 1, "2", 'three', 4.5]
var b = new Array()
for (i = 0; i < a.length; i++)
    b[i] = a[i]
```

- **other array methods: sort, shift, join, reverse, push, pop, ...**
- **libraries for math, strings, reg exprs, dialog boxes, date/time, ...**

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 action="http://www.google.com/search"
    name=srch>
<input type=text size=25 name=q
    id=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.gif" onMouseover='src="frown.gif"'
                      onMouseout='src="smiley.gif"'>
```

- 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
- **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 of most elements 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>
```

More 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 bitmap graphics)
  - HTML5 (next version, to help replace Flash, Silverlight, etc.)
- **XUL (XML user interface language)**
  - built from CSS, Javascript, DOM
  - only in Firefox?
  - portable definition of common widgets like buttons
- **browser plug-ins and extensions**
  - Firebug
  - Greasemonkey
- ...