Web interfaces

- Javascript
- HTML
- DOM
- CSS
- XMLHttpRequest
- Ajax

Reprise of HTTP

- What happens when you click on a URL?
- client sends request:
  
  ```
  GET url HTTP/1.0
  (blank line)
  ```

- server returns
  
  ```
  header info
  (blank line)
  HTML
  ```

  - since server returns the text, it can be created as needed
  - can contain encoded material of many different types (MIME)

- URL format
  
  ```
  service://hostname/filename?other_stuff
  ```

  filename?other_stuff part can encode
  
  - data values from client (forms)
  - request to run a program on server (cgi-bin)
HTML form

```html
<html>
<body>
<form method="GET"
    action="http://campuscgi.princeton.edu/~bwk/hello1.cgi">
    <input type="submit" value="hello">
</form>
</body>
</html>
```

• form data is URL-encoded in query string (GET)
  or to server's stdin (POST)
• limited interaction on client side
• requires synchronous exchange with server
  - potentially slow, client blocks waiting for response
• requires recreating entire page with whatever comes back
  - even if it’s identical to current content

• how can we make web interfaces more interactive and responsive?

Javascript

• very widely used programming language
• all browsers support it (though not identically)
• usually enabled (though not always)

• simple scripting language
  - C/Java-like syntax
  - about the level of Awk
  - very weakly typed
    - basic data types: double, bool, string, array, object
  - object-oriented

• runs inside browser
  <script> javascript program </script>
  <script src="url"></script>
  <sometag onSomeEvent='javascript code'>

• can catch events from mouse, keyboard, ...
• can access browser's object interface
  - window object
  - document object (DOM == document object model)
• can create original page and alter it later
Javascript on a page or two

- case sensitive
- semicolons or newline as statement terminators
- \// or /"..."/ comments
- var x to declare variable
  - scope is either global or current function
- double, bool, 'string' or "string" with \ escapes
  - null for undefined value
- operators, expressions, and control flow
  are like C or Java, sort of
  - for (v in obj)...
  - try(...) catch(...) finally(...)
- user-defined functions
  function sum(x, y) { return x + y; }
- arrays are sort of quasi 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, ...

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
Sorting (the hard way)

```javascript
var name, i = 0, j, temp
var names = new Array()

name = prompt("Enter new name")
while (name != "") {
    names[names.length] = name
    name = prompt("Enter new name")
}

for (i = 0; i < names.length-1; i++) {
    for (j = i+1; j < names.length; j++) {
        if (names[i] > names[j]) {
            temp = names[i]
            names[i] = names[j]
            names[j] = temp
        }
    }
}

s = names[0]
for (i = 1; i < names.length; i++)
    s += 
    alert(s)

• the easy way:
    names.sort()
    alert(names.join("\n"))
```

Javascript library

• math
  - sqrt, max, min, random, ...
• string
  - searching, substring, case conversion,
  - convert to HTML, ...
• regular expressions
  - about the same as Perl
• date/time
  - current time, elapsed time, conversions
• ...

Javascript objects

• objects are associative arrays
  - associate names with properties
  - name of property is the subscript

• can define your own objects
  - including inheritance

• can create anonymous objects
  \[ \text{var o = \{ x:1, y:2, z:hello\} }; \]

• browser environment includes objects like window and document

DOC: Document Object Model

• a web page in HTML (or XHTML) is structured data
  - XHTML is a tag set for HTML
• the document object model (DOM) is a representation of this hierarchy

• DOM methods, properties and events are accessible from Javascript
  - usually in `<form>` tag for buttons, text, etc.
  - can also appear in other tags, images, ...
  - event handling code can be attached to tags as attributes

• window methods and properties
  - `alert(msg), prompt(msg), ...
  - `open(uri)
  - `size, position, scrolling, ...
  - `history, status bar, ...
  - `document
Embedding Javascript

- in a form:
  
  <form>
  <input type=button value="Hit me"
     onClick='alert("Ouch! That hurt.")'>
  <input type=text name=url size=30>
  <input type=button value="GO"
     onClick='window.open(url.value)'>
  <input type=button value="color it 
     onClick='document.bgColor=color.value'>
  <input type=text name=color
     value='type a color here'>
  <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="new.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)

p { font-family: "Garamond", serif; }

h2 { font-size: 110%; color: red;
    background: white; }

a:hover { text-decoration: none;
           color: #f0f; font-weight: bold }

- style property of most document entities can be set by Javascript

  <body id="body">
  <script>
    var b = document.getElementById("body")
    b.style.backgroundColor='lightyellow'
    b.style.fontFamily='Verdana'
    b.style.fontSize='72px'
    b.style.color='blue'
  </script>
  hello
XMLHttpRequest

- interactions between client and server are usually synchronous
  - so there can be significant delay
  - and page has to be redrawn

- XMLHttpRequest provides **asynchronous** communication with server

- used in Google Suggest and Google Maps
  - also Orkut, Gmail, Flickr, A9 (it is said)

- "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 for an oldish idea)
  - XHTML + 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)

Google Suggest in microcosm

```html
<body>
  <form>
    Search:
    <input type="text" id="pat" onkeyup='geturl(pat.value); return true;' >
  </form>
</body>
```
Basic structure

```javascript
var req;

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

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

function geturl() {
  url = 'http://www.cs.princeton.edu/~bwk/echo.cgi';
  loadXMLDoc(url); // loading is asynchronous
}

function show(s) {  // show whatever came back
  var e = document.getElementsByTagName("P")[0]
  e.firstChild.nodeValue = s
}
```

XMLHttpRequest methods/properties

![XMLHttpRequest methods/properties table]

```javascript
XMLHTTP methods/properties
```

![XMLHTTP methods/properties table]
Assessment

- **potential advantages**
  - can be much more responsive (cf Google maps)
  - can offload work from server to client

- **potential negatives**
  - Javascript has to be enabled
  - Javascript is not a great language
  - asynchronous code can be hard to write
  - DOM is very awkward
  - mechanism not yet fully standardized
  - Javascript code is exposed to client

- **what next?**
  - better libraries for XML, DOM?
  - better tools and languages for programming?
  - better standardization?