

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>
```
- **form data sent to server for processing**
- **limited interaction on client side**
- **synchronous exchange with server**
 - potentially slow: client blocks waiting for response
- **recreates entire page with what comes back**
 - even if it's 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

Javascript

- **client-side scripting language**
 - 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
 - created by Brendan Eich, 1995 (at Netscape)
- **all browsers support it**
 - reasonably well standardized (HTML is much less standardized)
 - usually enabled

```
<script> javascript code </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) entities on page
- **can create original page and alter it later**

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, ...**

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>
```

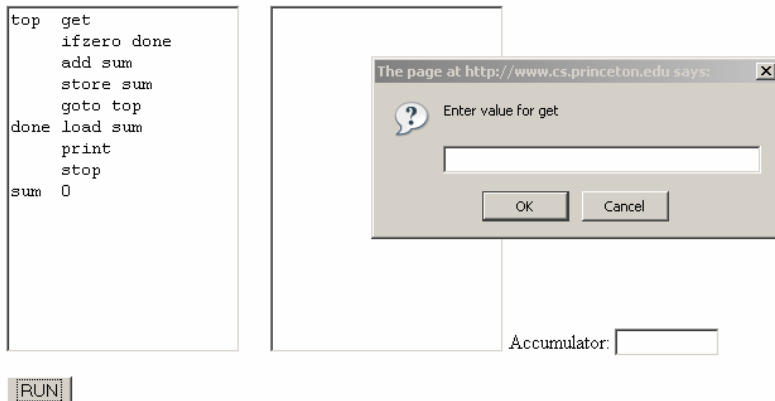
- **needs parseInt or parseFloat to coerce string value to a number**

Toy simulator

COS 109 Toy Machine Simulator

(You must have Javascript enabled.) Type your program in the left window. **Labels must start in the first column and operators like GET or ADD must start anywhere but the first column (i.e., there must be a space or tab before them).** The simulator does not distinguish upper case from lower case, but is otherwise not robust, so be sure to spell instructions correctly and format code carefully.

Push RUN to run your program. A dialog box will appear when a GET is executed, and output from PRINT will appear in the right window. The simulator will stop if you Cancel a GET or don't enter anything.



Syntax reminder

get	get a number from keyboard into accumulator
print	print contents of accumulator
load Val	load accumulator with Val (Val unchanged)
store M	store contents of accumulator into memory location M (accumulator unchanged)
add Val	add Val to contents of accumulator (Val unchanged)
sub Val	subtract Val from contents of accumulator (Val unchanged)
goto L	go to instruction labeled L
ifpos L	go to instruction labeled L if accumulator is >= zero
ifzero L	go to instruction labeled L if accumulator is zero
stop	stop running
Num	initialize this memory location to numeric value Num (once, before program)

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 colo
  <input type=button value='make it white'
    onclick='document.bgColor="white" '>
</form>
```

- **in a tag**

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

- **on an image**

```

```

- **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, ...

```
p { font-family: "Garamond", serif; }
h2 { font-size: 110%; color: red;
      background: white; }
a:hover { text-decoration: none;
          color: #f0f; font-weight: bold }
```

- 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='72px'
b.style.color='blue'
</script>
hello
```

CSS dynamic positioning

- "style" attributes for positioning elements
 - 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

Other HTML stuff

- **specialized markups**
 - SVG (scalable vector graphics)
 - Canvas Tags (scriptable bitmap graphics)
 - MathML, ...
- **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
- ...