CSCI 1320
Creating Modern Web Applications
Lecture 5: JavaScript
Experience Reports

• What page did you look at and what types of things did you find that were dynamic?
• What did you find that you think should be dynamic?
JavaScript & Tracks

• **Designer**
  • What JavaScript can be used for
  • When it should be used on a web page

• **Concentrator**
  • How to use JavaScript
  • Common errors
  • How to write and debug JavaScript code

2/2/18

Lecture 5: JavaScript
Static versus Dynamic Pages

• What does dynamic mean
• Many good application user interfaces are dynamic
  • Examples
• Web Pages are inherently static
  • HTTP model: action replaces the page
• Web applications require dynamics
  • Can these be done with pure HTML?
HTML is Basically Static

- Provides a description of the page, **not what to do with it**
  - Dynamics from built-in widgets (forms)
    - Clicking on submit then causes a new page request
    - With name-values pairs for the widgets as part of the URL or post
    - Result is a page that REPLACES the current page
  - Dynamics from CSS tags
  - Limited interaction
- **Is this sufficient for a web application?**
How to Allow Interactivity

• Plugins
  • Code (library) that is loaded into the browser
    • Using a somewhat standard API
    • Browser and platform specific
  • Introduce security and other problems
    • People are told not to install these

• Applets
  • Java programs downloaded and run by the browser
    • Using a standard interface
  • Introduce security and other problems
    • Java runs in a sandbox (you hope)
    • People are told not to enable these
How to Allow Interactivity

- Extend HTML into a programming language
  - Historically different languages tried
    - VBScript, JavaScript, others
    - Hence the SCRIPT tag in HTML
  - CSS has some capabilities for interactivity
    - But it isn’t a programming language
  - Eventually **JavaScript** won
    - Officially ECMAScript (standardized)
  - Available in almost all browsers today
    - Can be disabled
Questions

Which of the following does JavaScript not support
A. Associative arrays
B. Indexed arrays
C. Creating objects
D. Object inheritance
E. Multiple threads of control

The JavaScript language is
A. Procedural
B. Functional
C. Object-oriented
JavaScript Example

• Sumcompute.html
  • Show in operation
  • Lets look at how it works
What is JavaScript

• Type-less (dynamically typed) Scripting Language
  • Data is typed dynamically (at run time) rather than statically
  • Language is interpreted rather than compiled (in theory)
• Complete with libraries
  • Library provides basic functions (string,...)
  • Library provides access to browser capabilities
• Automatically invoked by the browser
  • Notion of events, on-conditions
  • Reactive language
• Can be embedded in HTML or in separate files
JavaScript is Procedural

- Standard control constructs
  - Loops, conditionals, ...

- Functions and calls are the primary mechanisms
  - User-defined functions
  - Functions called by browser
  - Library functions
JavaScript is Functional

- Functions are first class objects
- Can be passed and used explicitly
- Lambda expressions and continuations

```javascript
function sum1(x, y) { return x + y; } // definition
function sum2(x, y) { return x + y; } // return value

var n = sum1(5, 5); assert(n == 10); // call

var sum2 = function(x, y) { return x + y; }; // 3 ways to define a
var sum3 = new Function("x","y","return x+y;"); // function
assert(sum1.toString() == "function sum1(x,y) { return x+y; }"); // reveals definition code, but
assert(sum3.toString() == "function sum(x,y) { return x+y; }"); // format varies

function sum() { // Dynamic arguments
  var retval = 0;
  for (var i = 0; i < arguments.length; i++) {
    retval += arguments[i];
  }
  return retval;
}
assert(sum(1, 2) === 3);
assert(sum(1, 2, 3, 4, 5) === 15);
```
JavaScript is Object-Oriented

- Objects with fields and methods
- Prototype-based, not class based
  - new Object(), {}
  - new Type()
    - Type is a function, not a class
- Latest JavaScript has real classes
JavaScript Declarations

• No types => No declarations
• Except you have scopes
  • Global scope
  • Function scope
  • No local scopes
  • But function scopes nest
• Variables are global by default
  • Except for parameters
  • Except for variables declared using `var` in a function
    • Good practice: declare all variables
JavaScript Data Types

- Numeric types (int, double)
- Booleans (true, false) \{0, NaN, "", null, undefined\}
- null, undefined
- Arrays
  - Indexed (Arrays)
  - Associative arrays (Objects)
- Strings
- Regular Expressions (/pattern/g)
- Functions
- Objects
  - Field-value pairs with some inheritance
  - Values can be functions (methods)
  - Associative arrays
JavaScript Strings

- Can use single or double quotes
  - Backslash escapes
- JavaScript is designed somewhat for string processing
- String equality (s == “hello”)
- String concatenation (“hello” + “ ” + “world”)
- Other string functions
  - indexOf, split, substring, charAt, toUpperCase, toLowerCase
JavaScript Regular Expressions

- Regular expressions are useful in web applications
  - Checking formats, values
  - Advanced string processing (find/replace)
- `var x = /pattern/mods`
- `var re = /ab+c/;`
- `var re = /\bt[a-z]*\b/i;`
JavaScript Control Constructs

- for (init; test; update) { ... }
- if (test) { ... } else { ... }
- while (test) { ... }
- break
- continue
- for (var x in expr) { ... }
  - Expr is an object
  - x is the fields (indices) of the object
- try ... catch
JavaScript Functions

• `function name(arg, arg, ...) { ... }`
  - No argument matching
  - return value
  - `name = function(args) { ... }`

• **Functions are first-class objects**
  - Can be assigned to variables
  - Can be passed as arguments
  - Can be used as values
JavaScript Scopes

• Global scope and function (local) scope only
  • Variables are global unless otherwise stated
  • `var x;` declares a variable to be local if in function
    • Can occur anywhere in the scope (& multiple times)
  • parameters are local
  • Function scopes can nest with nested functions

• Many JavaScript problems are scope errors
JavaScript Objects

- An object is a dynamic mapping of fields to values
  - var x = new Object(); var x = { }
  - x.y = 5, x[“y”] = 5
  - x.plusone = function() { return x.y + 1; }
    - x.plusone() == 6
  - for (var x in object) { print x,object[x]; }
- Objects can be defined explicitly
  - function Type(a) {
    - this.field = a;
    - this.method = function() { return this.field+a; }; }
  - var x = new Type(5);
- Objects can be defined incrementally
  - Type.prototype.method = function() ...
Arrays are indexed
- x = [1, 2, “a”, 5]
- x[0] = 1; x[1] = 2;

Should be used as such
- Not associative
  - A[“hello”] works but not the way you expect
  - Why?
- Do not use for(var x in ARRAY)
  - Might not do what you expect
  - Missing elements, extra elements
JavaScript in the Browser

• Designed for interaction
  • JavaScript code is typically not running all the time
    • Invoked when something happens
  • What might that something be?

• Event Types
  • onLoad, onUnload (of page, frame, ...)
  • On widget value change; form submit; mouse over; ... 
  • On almost any possible event you may want to trigger on

• As part of HTML, can specify the event handler
  • onXxx='expression'
  • Expression is typically a call to a user-defined function
JavaScript and Threads

- **JavaScript is single-threaded**
  - Runs as a single thread in the browser too
    - Can hold up other things
  - Considered part of page loading or UI handling
    - The browser isn’t doing something else when your JavaScript is running
What Dynamics Can JavaScript Do?

- Check the values of fields before submission
- Compute new values as part of submission
  - E.g. encrypt a password
- Edit values in various ways
- Add values to other fields
- Establish timers and simple animations
- Draw on HTML5 canvases
- Modify parts of the page
JavaScript Debugging

• It is useful to be able to debug your JavaScript
• Most browsers today include a JavaScript debugger
  • Firebug as part of firefox
  • IE developer tools
  • Chrome developer tools
  • Safari developer tools
• Facilities
  • Set breakpoints, examine values
  • **Learn to use it**
  • Before you ask TAs what’s wrong
JavaScript Example

• Sumcompute.html
  • What was the problem
  • Let's look at how it works
JavaScript has its Quirks

- Objects are very different from Java/C++
- Functions are first class objects
- Scopes are defined based on `var` statements
  - Globally, not in-order, no local scopes
- Automatic type conversion
- Strict versus non-strict equality testing
- `eval` function
- Semicolons are optional if unambiguous
- Timeouts (`setTimeout` and `setInterval`)
- Read up on the language (prelab)
What Else Would You Like to Do

- Change the page, animate things
- This can require extensive computation
- Next Time
Next Time

• DOM manipulation
• Assignment 1 is out
• Lab 1 is due before next lecture
• Project preferences in by midnight
• Homework:
  • PreLab 2: to familiarize yourself with JavaScript
• Lecture Preparation:
  • Take a web page of your choosing and see what happens when you change the size of the browser, both in terms of the aspect ratio and in terms of absolute size. Is this what you would like/think should happen. Come prepared to discuss.
JavaScript Demo

• Basic types
JavaScript Demo

- Objects and Arrays
A browser plug-in is:

A. A JavaScript file that can be downloaded as part of a page.
B. A library that becomes part of the browser once downloaded and accepted by the user.
C. A technique that allows you to play a video directly in the browser.
D. A Java program that is run inside the browser.
E. A file that is loaded into the web server to handle special access for a web application.