**CS1320 Creating Modern Web and Mobile Applications** Lecture 5:

JavaScript

### JavaScript & Tracks

#### Designer

- What JavaScript can be used for
- When it should be used on a web page
- How and when to incorporate interaction into your designs
  - What is possible, easy, ...

#### Concentrator

- How to use JavaScript
- Common errors
- How to write and debug JavaScript code
- What are the important features of the language



# **Static versus Dynamic Pages**

- What does dynamic mean
- Most good application user interfaces are dynamic

• Examples

• Web Pages are inherently static

• HTTP model: action replaces the whole pag

• 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 causes a new page request, not an action on the page
    - With name-values pairs for the widgets as part of the URL or post data
    - Result is a page that **REPLACES** the current page
  - Limited dynamics from CSS tags
  - o Limited interaction
- Is this sufficient for a web application?



# How to Allow Dynamic Interactivity

#### • Plugins

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

#### • Applets

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



# How to Allow Dynamic Interactivity

- Extend HTML into a programming language
  - o Historically different languages tried
    - VBScript, JavaScript, others
    - Hence the SCRIPT tag in HTML
  - o CSS has some capabilities for interactivity
    - But it isn't a programming language
  - Eventually JavaScript won
    - Officially ECMAScript (standardized); ES6 is the latest incarnation
  - o Available in almost all browsers today
    - Can be disabled



### 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)
- TypeScript typed JavaScript is becoming more common

#### Complete with libraries

- Libraries providing basic functionality (strings,...)
- o Libraries providing access to browser capabilities

#### Automatically invoked by the browser

- o Notion of events, on-conditions
- Reactive language
- Can be embedded in HTML or in separate files for web pages
- Used for other purposes as well
  - o Back end: Node.JS; Mobile Applications: NativeScript, React Native

nima. Por de avectares nubico na hacer referencia a nagon obje ejecutindo las minuccioner y evaluando los objetos premarante

var z + 21 var y + 101 var z + "3" document.write(ava)("z + y + z + 1"))



unjeto.toStrrng() LaBaddad de une métoda, comin var mesou + nove Arroy("Exoro", "Februar", "Has Come parahemes, metor er un objets, yne er posible comme document.orritu("Los metors nou; " + metors)

#### **JavaScript is Procedural**

- Standard control constructs
  - Loops, conditionals, ...
- Functions and calls are the primary
  - o User-defined functions
  - o Functions called by browser
  - Library functions
- Modules provide high-level organization



Procedures

#### **JavaScript is Functional**

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

	Function function zed() { return 0; }			
	<pre>function sum(x,y) { // definition return x+y; // return value</pre>			
	<pre> / var n=sum(5,5); assert(n == 10); // call </pre>			
<pre>function sum1(x,y) { return x+y }; // 3 way var sum2=function(x,y) { return x+y; } // de var sum3=new Function("x","y","return x+y;); // fun assert(sum1.toString() == // reveals definition codd "function sum1(x,y) { return x+y; }"; // c format</pre>				
	<pre>function sumx() { // Dynamic arguments var retval=0; for (var i=0; i &lt; arguments.length; i++) { retval+=arguments[i]; } return retval;</pre>			
	<pre>     assert(sumx(1,2) == 3);     assert(sumx(1,2,3,4,5) == 15); </pre>			

# **JavaScript is Object-Oriented**

- Objects with fields and methods
- Prototype-based, not class-based
- new Object(), { }, { x : 1, y: 3+4 }

• Dynamic set of fields and methods

new Type()

• Type is a function, not a class

- o Use of **this** inside its methods refers to the object
- Latest JavaScript has Java-like classes with inheritance, etc.
  - Syntactically, not internally

Cake.prototype	
yellowCakeproto_	
Cake.prototype	E (crust :"chock

#### **JavaScript Declarations**

- No types => No declarations
- Except you have scopes
  - o Global scope
  - Function scope
  - Local scopes (sometimes)
  - But function scopes nest

#### • Variables are global by default

- o Except for parameters
- Except for variables declared using **var** or **let** or **const** in a function
  - Good practice: declare all variables
  - Good practice: use let or const, not var

#### Random Access Memory.

RAM provides the computer program with memory reserved for variable definitions and other data storage. All JavaScript variables are stored in computer memory.



Variables are created and stored inside the computer processor.

When a variable is defined, depending on the type of the computer processor, a number of bytes will be allocated to store the value specified by the programmer.

In this case, enough memory to store the integer value 1 will be allocated.

On a 32-bit processor, 32 bits of space will be allocated to store this number. In binary format, it looks like this:

### JavaScript Data Types

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



Lecture 5: JavaScript

#### **JavaScript Strings**

- Can use single or double quotes
  - o 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
  - o endsWith, startsWith, contains



# **JavaScript Templated Strings**

**`JavaScript \${templateStrings}`** 

- Use backquote (`) as delimiter
- Can span multiple lines (new line included in the string)
- Can have embedded expressions
  - o \${expression}
  - Replaced with the string value of the expression
  - Replaces concatenation operations and complex expressions

### JavaScript Regular Expressions

- Regular expressions are useful in web applications
  - o Checking formats, values
  - Advanced string processing (find/replace)
- Iet x = /pattern/mods
  - o let re = /ab+c/;
  - $\circ$  let re = /\bt[a-z]\*\b/i;
- Regular expression are a basic prin



Anchors		Sampl	e Patterns	
*	Start of line +	([A-Za	-z0-9-]+)	
1A.	Start of string +	(\d(1,	2)\/\d{1,2}\/\d{4})	
\$	End of line +	([^\s]+(?=\.(jpg[gd[png)]\_2) (^[1-9]{1}\$!^[1-4]{1}[0-9]{1} (#?([A-Fa-f0-9]){3})(([A-Fa-f0-9] ([?=,*\d)(?=,*[a-2])(?=,*[A-2]).		
\Z	End of string +			
\b	Word boundary +			
18	Not word boundary +			
14	Start of word			
B	End of word			
		(\w+0	[s-zA-Z_]+7\.[s-zA-Z]{2,6	
Character Classes		(\<(/7[^\>]+)\>)		
10	Control character	and the second second	These patterns are intended	
10	White snace	Hote	Please use with caution and	

### **JavaScript Control Constructs**

- if (test) { ... } else { ... }
- while (test) { ... }
- switch (expr) { case: ... }
- break, continue
- for (init; test; update) { ... }
- for (let x in expr) { ... }
  - Expr is an object
  - o x is the fields (indices) of the object
  - Can use const or let
- for (const x of expr) { ... }
  - Expr is an Iterable (Array, String, Map, Set, NodeList, ... )
- try ... catch



#### **JavaScript Functions**

- function name(arg,arg,...) { ... }
  - No argument matching (type or number)
  - return value
  - o name = function(args) { ... }
  - Default argument values: arg = value
  - Varargs (arg,arg, ...rest)
- (arg,arg) => expression
  - o (arg,arg) => { statements }
- Functions are first-class objects
  - Can be assigned to variables
  - Can be passed as arguments
  - Can be used as values



### **JavaScript Scopes**

Global, function, and local scopes



- 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)
  - let x; declares a variable to be in local scope
  - const x; declares a non-changeable variable in the local scope
  - o 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
  - o let x = new Object(); let x = { }
  - x.y = 5, x["y"] = 5
  - o x.plusone = function() { return x.y + 1; }
    - x.plusone() == 6
  - o for (let x in object) { print x,object[x]; }
- Objects can be defined explicitly
  - o function Type(a) {
    - this.field = a;
    - this.method = function() { return this.field+a; }; }
  - o var x = new Type(5);
- Objects can be defined incrementally
  - Type.prototype.method = function() ...



## **JavaScript Arrays**

- Arrays are indexed (numerically)
  - o x = [ 1,2,"a",5]
  - o x[0] = 1; x[1] = 2;
  - o x[2]= "a"; x[3] = 5;

#### Should be used as such

- o Not associative
  - A["hello"] works but not the way you expect
  - Why?
- Do **not** use for(let x in ARRAY)
  - Might not do what you expect
  - Missing elements, extra elements
- You can use for(let i = 0; I < ARRAY.length; ++i) or for(let x of ARRAY)

# [0]-0; #111-0;

BLOG ARCHIVES		Hummingy, Agenii 1 2013 11:12:38 AM
The Javascript Reference Seri	PDF Available <sup>9</sup> Postering Javascrpt Annes (227.407 Intes)	
<ul> <li>Essential Jacascript — A Jacascript Totorial</li> <li>Jacascript Numbers</li> <li>Jacascript Schuse</li> <li>Jacascript Anters</li> </ul>	<ul> <li>denserigt Einstlans</li> <li>BUD idenserigt Objects)</li> <li>denserigt Data</li> <li>data Ziber Books</li> </ul>	Article Index <sup>9</sup> Extraction <sup>9</sup> Creating A New Array <sup>9</sup> Extendedien <sup>9</sup> Creating A New Array <sup>9</sup> Scheme Data In An
Mastering Javascrip	Array <sup>9</sup> Nuiti-Dimensional Arrays	
Mastering Assuscript Arrays	<sup>4</sup> Javascript Arrays Are Passed By Reference	
Believe it or not, the very first version of Assascript ab for the oversight and modern Assascript Arrays are po examined data structures such as stacks and queues. If as well as introduce a few methic extensions.	* Javascript Amays Are Assigned by Reference * Possing Amays As Values * Amay Length	
Introduction		*Javascript Does Not Support Associative
An Array is an enumerated list of variables. It's a prog- replace this	Arroys *Arroy Hethods Balenesse	
11-0 12-0 12-0 12-0 12-0 12-0 12-0 12-0		* American concept; native 1[ value 2[ value * American every(function) * American Bloc(function) * American for flands/function} * American for flands/function} * American for flands/function}
_ with this		startindes])

startindex))

### 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

- o onXXX='expression'
- Expression is typically a function or a call to a user-defined function

#### Mac OS X browser JavaScript performance

milliseconds



average of three runs of Sunspider 0.9 JavaScript benchmark shorter bars are better

### JavaScript and Threads

- JavaScript is single-threaded
  - o 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
- Promises provide language-level background processing
  - We'll cover these next week



### What Dynamics Can JavaScript Do?

- Check the values of fields before submission
- Compute new values as part of subm

• E.g. encrypt a password

- Edit values in various ways
- Add values to other fields
- Establish timers and simple animatio
- 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
  - o IE developer tools
  - Chrome developer tools
  - o Safari developer tools

#### Facilities

- Set breakpoints, examine values
- Learn to use it

• Before you ask TAs what's wrong

O Developer Tools - http://localhost:18437/					
Elements	Resources Network Scripts Timeline Profiles	Audits Console Search Scripts			
- ◄ ►	DefaultJs?v=f0.QL7O0vPVHk055 🔨	▶ ? † † <b>r</b> /≥	Paus	sed	
1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930	<pre>iv (arguments.rengtn) {     var n = this.pushStac     return n.push.apply(r     } },remove: function(n, t) {     for (var u = 0, r; (r = t         (ln    i.filter(n, [r]         return this },empty: function() {     for (var t = 0, n; (n = t</pre>	<pre>&gt; Watch Expressions &gt; Call Stack &gt; Scope Variables V Breakpoints V perfsurf.js87 start: start, V DOM Breakpoints V body.ui-mobile-viewport</pre>	+ C		
1931	while (n.firstChild)	Subtree Modified			
1933 1934 1935 1936	<pre>n.removeLnld(n.t } return this },clone: function(n, t) { return p = p = p = pull 2 11</pre>	AHK Breakpoints     Any XHR     Vent Listener Breakpoints     Wethered	+		
1936 1937 1938 1939 1940 1941 1942 1943 1944 1944	<pre>return n = n == null ? !! return i.clone(this, }) },html: function(n) {     if (n == t)         return this[0] &amp;&amp; thi     if (typeof n != "string"     i.sFunction(n) ? thi         +</pre>	<pre>&gt; Keyboard  &gt; Keyboard  &gt; Mouse  </pre> Click  dblclick  mousedown  mouseup  mouseover  mousewove	4		
	Q 0 {} /		▲ 1	\$	

### JavaScript Example

- Sumcompute.html
  - What was the problem
  - Lets look at how it works

### JavaScript has its Quirks

- Objects are very different from Java/C++
  - Newer versions have Java-like classes however
- Functions are first class objects
- Function scopes are defined based on var staten
   Globally, not in-order,
- 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 computatior
- Next Time

JavaScript and Ajax AJAX 🚆 JavaScript

#### **Next Time**

- DOM manipulation
- Assignment 1 is out
- Lab 1 is due before next lecture
- Project preferences due by midnight
- Homework:

• PreLab 2: to familiarize yourself with JavaScript



#### JavaScript Demo

• Basic types



#### JavaScript Demo

• Objects and Arrays



### Question

#### 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.

#### Questions

#### The JavaScript language is

- I. Procedural
- II. Functional
- III. Object-oriented
- A. I
- B. 111
- C. I and II
- D. I and III
- E. I, II and III

#### **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?