

G22.3033-002 Scripting Languages

7/3/2008
Web Applications
Databases

© Martin Hirzel G22.3033-002 NYU 7/3/2008 1

Outline

- Databases
- Stateful Web Applications
- Document Object Model
- AJAX

© Martin Hirzel G22.3033-002 NYU 7/3/2008 2

Soap-box

Why use a Database?

- Store data for months or years
 - Database may live longer than the web application(s) that you write for it
- Why not just use simple ad-hoc files?
 - Database remains consistent in the presence of multiple concurrent accesses
 - Database scales better when there is a lot of data, or a lot of accesses
 - Don't reinvent the wheel

© Martin Hirzel G22.3033-002 NYU 7/3/2008 3

Concepts

Relational Databases

- Database = collection of tables
- Table = relation = set of rows w/ same columns
- Row = tuple = one value per column
- Column = attribute = name+primitive type
- Only store primitive values, never nest tables

Recipe				Cup2G		Volume		Weight	
rec	ing	qty	unit	ing	g	unit	cup	unit	lb
cake	flour	2.5	cup	flour	110	cup	1	lb	1
cake	milk	3	tbsp	sugar	225	tbsp	16	oz	16
gravy	salt	2	tsp	butter	225	tsp	48	g	453
gravy	sugar	10	g			ml	236		

© Martin Hirzel G22.3033-002 NYU 7/3/2008 4

SQL

Types

Most database products have additional primitive types.

© Martin Hirzel G22.3033-002 NYU 7/3/2008 5

SQL

About SQL

- Structured Query Language
 - Query information from relational database
 - Declarative: describe what information to find, not how to find it
- SQL consists of two parts
 - DDL = Data Definition Language
 - DML = Data Manipulation Language
- Each database product (sqlite, MySQL, Oracle, DB2, ...) has own SQL dialect
 - We use sqlite in this course

© Martin Hirzel G22.3033-002 NYU 7/3/2008 6

sqlite

How to Write + Run Code

- From PHP script
→ later in today's lecture
- By hand, from command line

```
doowop1> /usr/bin/sqlite3 test.db
SQLite version 3.3.6
Enter ".help" for instructions
sqlite> CREATE TABLE RecIng(rec VARCHAR(50), ing VARCHAR(50),
qty FLOAT, unit VARCHAR(10), PRIMARY KEY(rec, ing));
sqlite> INSERT INTO RecIng VALUES('cake','flour',2.5,'cup');
sqlite> INSERT INTO RecIng VALUES('cake','milk',3,'tbsp');
sqlite> INSERT INTO RecIng VALUES('gravy','salt',2,'tsp');
sqlite> SELECT ing, qty FROM RecIng WHERE rec = 'cake';
flour|2.5
milk|3.0
sqlite>
```

RecIng			
rec	ing	qty	unit
cake	flour	2.5	cup
cake	milk	3	tbsp
gravy	salt	2	tsp
gravy	sugar	10	g

WARNING: On CIMS machines, web server uses `sqlite2` ≠ `sqlite3`

© Martin Hirzel G22.3033-002 NYU 7/3/2008 7

sqlite

Create Table Statement

```
CREATE TABLE RecIng(rec VARCHAR(50), ing VARCHAR(50),
qty FLOAT, unit VARCHAR(10), PRIMARY KEY(rec, ing));

createTable ::=
CREATE [TEMP|TEMPORARY] TABLE (newTable | derivedTable)
newTable ::=
[IF NOT EXISTS] [id .] id ( column* (, tableConstraint)* )
derivedTable ::= [id .] id AS select
column ::= id [type] [(CONSTRAINT id) columnConstraint]*
columnConstraint ::=
NOT NULL [conflict] | UNIQUE [conflict]
| PRIMARY KEY [sortOrder] [conflict] [AUTOINCREMENT]
| CHECK (expr) | DEFAULT expr | COLLATE collationName
tableConstraint ::=
PRIMARY KEY (id*) [conflict]
| UNIQUE (id*) [conflict] | CHECK (expr)
conflict ::= ON CONFLICT conflictAlgorithm
conflictAlgorithm ::= ROLLBACK|ABORT|FAIL|IGNORE|REPLACE
```

© Martin Hirzel G22.3033-002 NYU 7/3/2008 8

sqlite

Insert Statement

```
INSERT INTO RecIng VALUES('cake','flour',2.5,'cup');
INSERT INTO RecIng VALUES('cake','milk',3,'tbsp');
INSERT INTO RecIng VALUES('gravy','salt',2,'tsp');
```

```
insert ::=
INSERT [OR conflictAlgorithm] INTO [id .] id [(id*)] VALUES (expr*)
insert ::=
INSERT [OR conflictAlgorithm] INTO [id .] id [(id*)] select
conflictAlgorithm ::= ROLLBACK|ABORT|FAIL|IGNORE|REPLACE
```

© Martin Hirzel G22.3033-002 NYU 7/3/2008 9

sqlite

Select Statement

```
select ::=
SELECT [ALL | DISTINCT] result*
[FROM table (joinOp table joinArgs)*]
[WHERE expr]
[GROUP BY expr*]
[HAVING expr]
(compoundOp select)*
[ORDER BY (expr [sortOrder])*]
[LIMIT integer [(OFFSET | ,) integer]]
result ::= * | tableName .* | expr [[AS] id]
table ::= tableName [AS alias] | (select) [AS alias]
joinOp ::= , |
[NATURAL] [LEFT|RIGHT|FULL] [OUTER|INNER|CROSS] JOIN
joinArgs ::= [ON expr] [USING (id*)]
sortOrder ::= [COLLATE collationName] [ASC|DESC]
compoundOp ::= UNION | UNION ALL | INTERSECT | EXCEPT
```

```
SELECT ing, qty
FROM RecIng
WHERE rec = 'cake';
```

© Martin Hirzel G22.3033-002 NYU 7/3/2008 10

sqlite

List of sqlite Statements

<u>Data Definition Language (DDL)</u>	<u>Data Manipulation Language (DML)</u>
• (CREATE ALTER DROP) TABLE	• INSERT
• (CREATE DROP) INDEX	• SELECT
• (CREATE DROP) TRIGGER	• UPDATE
• (CREATE DROP) VIEW	• REPLACE
• CREATE VIRTUAL TABLE	• DELETE
• ATTACH DATABASE	• BEGIN TRANSACTION
• DETACH DATABASE	• COMMIT TRANSACTION
• ANALYZE	• ROLLBACK TRANSACTION
• REINDEX	• END TRANSACTION
• VACUUM	• EXPLAIN
	• PRAGMA

© Martin Hirzel G22.3033-002 NYU 7/3/2008 11

SQL

Lexical Peculiarities

- Case insensitive
- Commands end with semicolon (;)
- Single-line comments: --...
- Multi-line comment: /*...*/
- String literal: ' s '
 - Escape single quote (') in string with another single quote ('), not with backslash (\')
- Identifier: simple *id* or quoted "*id*"
 - Quoted identifier can contain any character
 - Quoted identifier and can be same as keyword

© Martin Hirzel G22.3033-002 NYU 7/3/2008 12

sqlite

Operators

~, +, ~, NOT	1		Negation
CASE [expr] (WHEN expr THEN expr)* [ELSE expr] END			Conditional
CAST (expr AS type)	1		Conversion
	2	L	String concat.
*, /, %	2	L	Multiplicative
+, -	2	L	Additive
<<, >>, &,	2	L	Bitwise
<, <=, >, >=	2		Comparison
=, ==, !=, <>, IN	2		Identity
expr [NOT] (LIKE GLOB REGEXP MATCH) expr [ESCAPE expr]	2/3		Pattern match
expr [NOT] BETWEEN expr AND expr	3		Comparison
AND, OR	2		Logic
[EXISTS] (select)	1		Query-in-expr

© Martin Hirzel G22.3033-002 NYU 7/3/2008 13

sqlite

Library Functions

- Aggregate: avg, count, group_concat, max, min, sum, total
- String: glob, length, like, lower, ltrim, quote, replace, rtrim, soundex, substr, trim, upper
- Number: abs, max, min, random, round
- Misc: coalesce, ifnull, hex, last_insert_rowid, load_extension, nullif, randomblob, sqlite_version, typeof, zeroblob
- Date+time: date, time, datetime, julianday, strftime

© Martin Hirzel G22.3033-002 NYU 7/3/2008 14

Reference

SQL Documentation

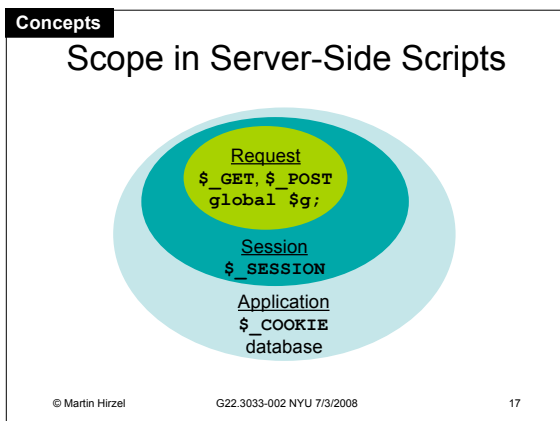
- Take a class on databases
- Read a standard databases text book
- sqlite: <http://www.sqlite.org>
- MySQL: <http://www.mysql.com>
- Tutorial: <http://www.w3schools.com/sql/default.asp>

© Martin Hirzel G22.3033-002 NYU 7/3/2008 15

Outline

- Databases
- Stateful Web Applications
- Document Object Model
- AJAX

© Martin Hirzel G22.3033-002 NYU 7/3/2008 16



PHP+SQL

Example from 5/22/2008, Revisited

WARNING: On CIMS machines, web server uses sqlite2 ≠ sqlite3

```

<?php
$d = sqlite_open("data/sqlite2", 0666, $err);
if ($err) { die($err); }
sqlite_query($d, "select * from T", SQLITE_BOTH, $err);
if ($err) {
    echo "table does not yet exist, creating it ...<br>";
    $q = "create table T(I integer, S char(10))";
    sqlite_query($d, $q, SQLITE_BOTH, $err);
    if ($err) { die($err); }
    sqlite_query($d, "insert into T values(0, 'n')");
}
$rows = sqlite_query($d, "select I from T where S='n'");
$row = sqlite_fetch_array($rows, SQLITE_BOTH);
echo "T[S=n][I]=" . $row['I'] . " . "; reload for ++<br>";
sqlite_query($d, "update T set I = I+1 where S='n'");
echo "delete data/sqlite2 to start over<br>";
?>
  
```

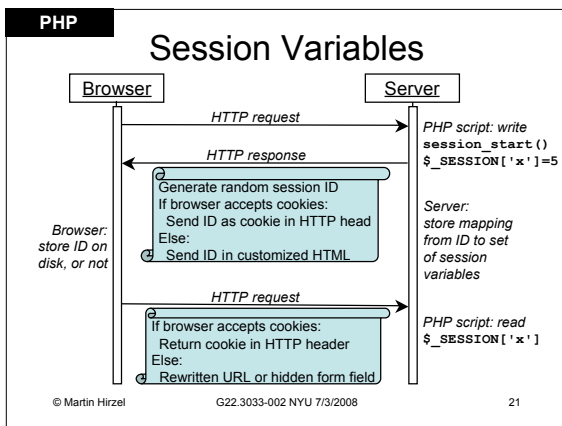
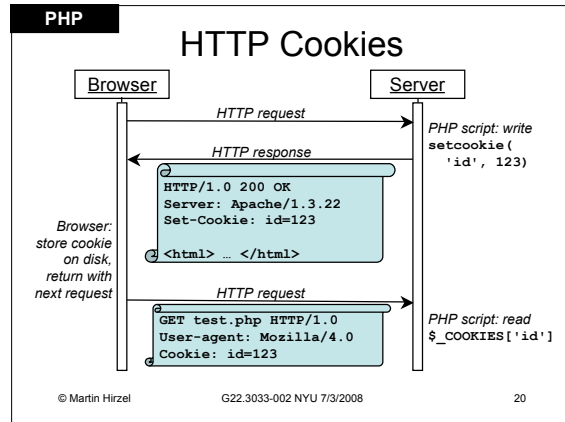
© Martin Hirzel G22.3033-002 NYU 7/3/2008 18

PHP+SQL

PHP Bindings for Database

- See also: <http://www.php.net/sqlite>
 - resource sqlite_open(string \$file [,int \$mode [,string &\$err]])
 - resource sqlite_query(resource \$db_handle, string \$query [,int \$result_type [,string &\$err]])
 - array sqlite_fetch_array(resource \$result [,int \$result_type [, bool \$decode_binary]])
 - void sqlite_close(resource \$db_handle)
 - ... and many more functions for PHP+sqlite
- PHP also has many other database bindings

© Martin Hirzel G22.3033-002 NYU 7/3/2008 19



Concepts

Cookies vs. Session Variables

Cookies	Session variables
Stored at client	Stored at server
May be rejected by browser	If browser rejects cookies, fall back on other mechanisms
More global scope	Local to page
Longer lifetime	Deleted after session

© Martin Hirzel G22.3033-002 NYU 7/3/2008 22

PHP

Common PHP Mistakes

- Compiler errors (blank page), permissions errors (access denied)
- Must set cookie (or call `session_start()`) before generating any HTML, so cookie can go in HTTP header, before payload
- Security errors
 - SQL injection: using unvalidated user input as part of database query
 - Cross-site scripting: using unvalidated user input as part of public forum HTML output

© Martin Hirzel G22.3033-002 NYU 7/3/2008 23

Outline

- Databases
- Stateful Web Applications
- Document Object Model
- AJAX

© Martin Hirzel G22.3033-002 NYU 7/3/2008 24

JavaScript

Library Functions

- Properties of the `Global` object
- Properties of objects stored in the Global object, for example, `Math`

© Martin Hirzel G22.3033-002 NYU 7/3/2008 25

JavaScript

The Global Object

<p><u>Value properties</u></p> <ul style="list-style-type: none"> • <code>NaN</code> • <code>Infinity</code> • <code>undefined</code> <p><u>Function properties</u></p> <ul style="list-style-type: none"> • <code>eval(x)</code> • <code>parseInt(string, radix)</code> • <code>parseFloat(string)</code> • <code>isNaN(string)</code> • <code>isFinite(number)</code> • ... 	<p><u>Constructor properties</u></p> <ul style="list-style-type: none"> • <code>Object(...)</code> • <code>Function(...)</code> • <code>Array(...)</code> • <code>String(...)</code> • <code>Boolean(...)</code> • <code>Number(...)</code> • <code>RegExp(...)</code> • <code>Error(...)</code> • ... <p><u>Other properties</u></p> <ul style="list-style-type: none"> • <code>Math</code>
--	--

© Martin Hirzel G22.3033-002 NYU 7/3/2008 26

JavaScript

The Array Constructor

<p><u>Called as constructor</u></p> <ul style="list-style-type: none"> • <code>new Array(len)</code> • <code>new Array(i0, i1, ...)</code> <p><u>Called as function</u></p> <ul style="list-style-type: none"> • <code>x=Array(...)</code> is same as <code>x=new Array(...)</code> <p><u>Properties of instances</u></p> <ul style="list-style-type: none"> • <code>length</code> <p><u>Properties of prototype</u></p> <ul style="list-style-type: none"> • <code>toString()</code> • <code>toLocaleString()</code> • <code>concat(a0, a1, ...)</code> 	<p><u>Properties of prototype (continued)</u></p> <ul style="list-style-type: none"> • <code>join(separator)</code> • <code>pop()</code> • <code>push(i0, ...)</code> • <code>reverse()</code> • <code>shift()</code> • <code>slice(start, end)</code> • <code>sort(compareFun)</code> • <code>splice(start, deleteCount, [i0, ...])</code> • <code>unshift([i0, ...])</code>
---	--

© Martin Hirzel G22.3033-002 NYU 7/3/2008 27

JavaScript

The Math Object

<p><u>Value properties</u></p> <ul style="list-style-type: none"> • <code>E</code> • <code>PI</code> • ... <p><u>Function properties</u></p> <ul style="list-style-type: none"> • <code>abs(x)</code> • <code>acos(x)</code> • <code>asin(x)</code> • <code>ceil(x)</code> • <code>cos(x)</code> • <code>exp(x)</code> • <code>floor(x)</code> 	<p><u>Function properties (continued)</u></p> <ul style="list-style-type: none"> • <code>log(x)</code> • <code>max(v1, ...)</code> • <code>min(v1, ...)</code> • <code>pow(x, y)</code> • <code>random()</code> • <code>round(x)</code> • <code>sin(x)</code> • <code>sqrt(x)</code> • <code>tan(x)</code> • ...
--	--

© Martin Hirzel G22.3033-002 NYU 7/3/2008 28

JavaScript

The RegExp Constructor

<p><u>Called as constructor</u></p> <ul style="list-style-type: none"> • <code>new RegExp(pattern, flags)</code> <p><u>Called as function</u></p> <ul style="list-style-type: none"> • <code>x=RegExp(...)</code> is same as <code>x=new RegExp(...)</code> <p><u>Properties of instances</u></p> <ul style="list-style-type: none"> • <code>source</code> • <code>global</code> • <code>ignoreCase</code> • <code>multiline</code> • <code>lastIndex</code> 	<p><u>Properties of RegExp prototype</u></p> <ul style="list-style-type: none"> • <code>exec(string)</code> • <code>test(string)</code> • <code>toString()</code> <p><u>Properties of String prototype</u></p> <ul style="list-style-type: none"> • <code>match(regex)</code> • <code>search(regex)</code> • ...
---	--

© Martin Hirzel G22.3033-002 NYU 7/3/2008 29

JavaScript

Example from 5/22/2008, Revisited

```
<html>
<head><title>Form validation example</title>
<script language="JavaScript">
function chk() {
  var v = document.myFm.num.value;
  if (v>=1 && v<=10) return true;
  alert("bad input " + v);
  return false; //abort commit
}
</script>
</head><body>
<form name="myFm" method="post" action="otherpage.htm">
Enter a number: <input size="4" type="text" name="num">
<input type="submit" value="OK" onClick="return chk()">
</form></body>
</html>
```

© Martin Hirzel G22.3033-002 NYU 7/3/2008 30

Concepts

Object Model

- Object-oriented API for embedded scripts
- We saw this on 5/29/2008 for VBA + PowerPoint

© Martin Hirzel G22.3033-002 NYU 7/3/2008 31

JavaScript

Client-Side JavaScript Object Model

- In web browser, the Global object is also a Window object
- DOM = Document Object Model
- DOM standards
 - Level-0 (legacy)
 - W3C Level-1 (1998)
 - W3C Level-2 (2000)
- Browsers implement incompatible DOMs

© Martin Hirzel G22.3033-002 NYU 7/3/2008 32

JavaScript

The Window Object

- All properties shown on slide "The Global Object"
- All properties shown under "Window" on slide "Client-side JavaScript Object Model"
- Feature testing example

```
function getSelectedText() {
    if (window.getSelection())
        return window.getSelection().toString();
    else if (document.getSelection())
        return document.getSelection().text;
    else if (document.selection)
        return document.selection.createRange().text;
}
```

© Martin Hirzel G22.3033-002 NYU 7/3/2008 33

JavaScript

Document Objects

<p><u>Value properties</u></p> <ul style="list-style-type: none"> • bgColor • cookie • domain • lastModified • location • referrer • images [] • forms [] • ... 	<p><u>Function properties</u></p> <ul style="list-style-type: none"> • open () • write (s0, ...) • writeln (s0, ...) • close () • createAttribute (name) • createComment (text) • createDocumentFragment () • createElement (tagName) • createTextNode (text) • getElementById (id) • getElementsByName (name) • getElementsByTagName (tag) • importNode (node, deep)
--	--

- One property for each named form
 - Each form has a property for each named element

© Martin Hirzel G22.3033-002 NYU 7/3/2008 34

JavaScript

Node Objects

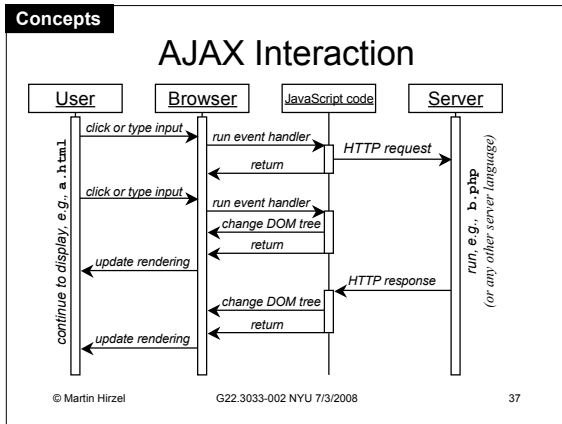
<p><u>Value properties</u></p> <ul style="list-style-type: none"> • innerHTML • nodeName • nodeType • nodeValue • childNodes [] • firstChild • lastChild • ownerDocument • parentNode • previousSibling • nextSibling • One property for each HTML attribute 	<p><u>Function properties</u></p> <ul style="list-style-type: none"> • hasChildNodes () • appendChild (newChild) • insertBefore (newChild, refChild) • removeChild (oldChild) • replaceChild (newChild, oldChild) • normalize () <p><u>Constants for nodeType</u></p> <table border="0"> <tr><td>1</td><td>ELEMENT_NODE</td></tr> <tr><td>2</td><td>ATTRIBUTE_NODE</td></tr> <tr><td>3</td><td>TEXT_NODE</td></tr> <tr><td>8</td><td>COMMENT_NODE</td></tr> <tr><td>9</td><td>DOCUMENT_NODE</td></tr> <tr><td>11</td><td>DOCUMENT_FRAGMENT</td></tr> </table>	1	ELEMENT_NODE	2	ATTRIBUTE_NODE	3	TEXT_NODE	8	COMMENT_NODE	9	DOCUMENT_NODE	11	DOCUMENT_FRAGMENT
1	ELEMENT_NODE												
2	ATTRIBUTE_NODE												
3	TEXT_NODE												
8	COMMENT_NODE												
9	DOCUMENT_NODE												
11	DOCUMENT_FRAGMENT												

© Martin Hirzel G22.3033-002 NYU 7/3/2008 35

Outline

- Databases
- Stateful Web Applications
- Document Object Model
- AJAX

© Martin Hirzel G22.3033-002 NYU 7/3/2008 36



Soap-box

Why use AJAX?

- AJAX = Asynchronous JavaScript and XML
- Asynchronous = non-blocking = send HTTP request to server without waiting for the response
- Advantages
 - Large images are not reloaded from server
 - User interface does not freeze up ("rich user experience", like non-web-applications)
 - Request-local client-side JavaScript state continues to be in scope
- See reading assignment from hw06

© Martin Hirzel G22.3033-002 NYU 7/3/2008 38

JavaScript

XMLHttpRequest Objects

<p>Value properties</p> <ul style="list-style-type: none"> • <code>onreadystatechange</code> //your event handler function • <code>readyState</code> <ul style="list-style-type: none"> //0 uninitialized //1 loading //2 loaded //3 interactive //4 complete • <code>responseText</code> //string • <code>responseXML</code> //Document • <code>status</code> //int: HTTP code • <code>statusText</code> //string: HTTP reason phrase 	<p>Function properties</p> <ul style="list-style-type: none"> • <code>abort()</code> • <code>getAllResponseHeaders()</code> • <code>getResponseHeader(parameterName)</code> • <code>open(method, url, /*bool*/ asynchronous)</code> • <code>send(content)</code> • <code>setRequestHeader(parameterName, parameterValue)</code>
---	--

© Martin Hirzel G22.3033-002 NYU 7/3/2008 39

JavaScript

AJAX Example: Files

```

doowop1:~$ pwd
/home/hirzel/public_html
doowop1:~$ ls -l a.html b.php c.js
-rw-r--r-- 1 hirzel 1088 387 Jul 2 20:19 a.html
-rw-r--r-- 1 hirzel 1088 38 Jul 2 20:14 b.php
-rw-r--r-- 1 hirzel 1088 753 Jul 2 20:19 c.js
doowop1:~$
    
```

© Martin Hirzel G22.3033-002 NYU 7/3/2008 40

JavaScript

AJAX Example: a.html

```

<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript" />
<script src="c.js"></script>
</head><body bgcolor="#00ff00">
Time: <span id="time">(please click button)</span>
<form name="in">
<input type="button" value="Toggle Color" onclick="localChange();">
<input type="button" value="Request Time" onclick="sendRequest();">
</form>
</body></html>
    
```

© Martin Hirzel G22.3033-002 NYU 7/3/2008 41

JavaScript

AJAX Example: b.php

```

<?php sleep(3); echo date('h:i:s') ?>
    
```

© Martin Hirzel G22.3033-002 NYU 7/3/2008 42

JavaScript**AJAX Example: c.js**

```
function localChange() {
  var blue = document.bgColor == '#0000ff'
  document.bgColor = blue ? '#00ff00' : '#0000ff';
}
function sendRequest() {
  document.getElementById('time').innerHTML = '(please wait)';
  var xhr = false;
  function handleResponse() {
    if (4 == xhr.readyState && 200 == xhr.status)
      document.getElementById('time').innerHTML = xhr.responseText;
  }
  try { xhr = new XMLHttpRequest(); } catch(e1) {
    try { xhr = new ActiveXObject("Msxml2.XMLHTTP"); } catch(e2) {
      try { xhr = new ActiveXObject("Microsoft.XMLHTTP"); } catch(e3) {
        alert('Could not create XMLHttpRequest object.');
```

© Martin Hirzel

G22.3033-002 NYU 7/3/2008

43

Soap-box**Evaluating JavaScript**Strengths

- Portability
 - Compared to VBScript in IE only
- Speed
 - Compared to making more server requests
- Small but powerful core language

Weaknesses

- Incompatible
- May be disabled in browser
- Too sophisticated
 - Closures, prototypes

© Martin Hirzel

G22.3033-002 NYU 7/3/2008

44

JavaScript**Common JavaScript Mistakes**

- If you test in just one browser, it probably will not work in other browsers
- Browser may not send server request if item in cache -> randomize URL
- JavaScript should enhance web pages, but web pages should also work without it
 - User may disallow scripts for security
 - Search engines won't index dynamic content
- User interface may be non-intuitive (back button, active elements, progress reports)

© Martin Hirzel

G22.3033-002 NYU 7/3/2008

45

JavaScript**Last Slide**

- Pick up graded quiz2 and homeworks
- See class page for point distribution
- Today's lecture
 - Web applications
 - Databases
 - Session state
 - Form validation
 - AJAX
- Next lecture
 - Guest lecturer: Marco Pistoia
 - Security for web languages
 - This will be part of exam / homework

© Martin Hirzel

G22.3033-002 NYU 7/3/2008

46