CS 5142
Scripting Languages

10/16/2015

Web Applications

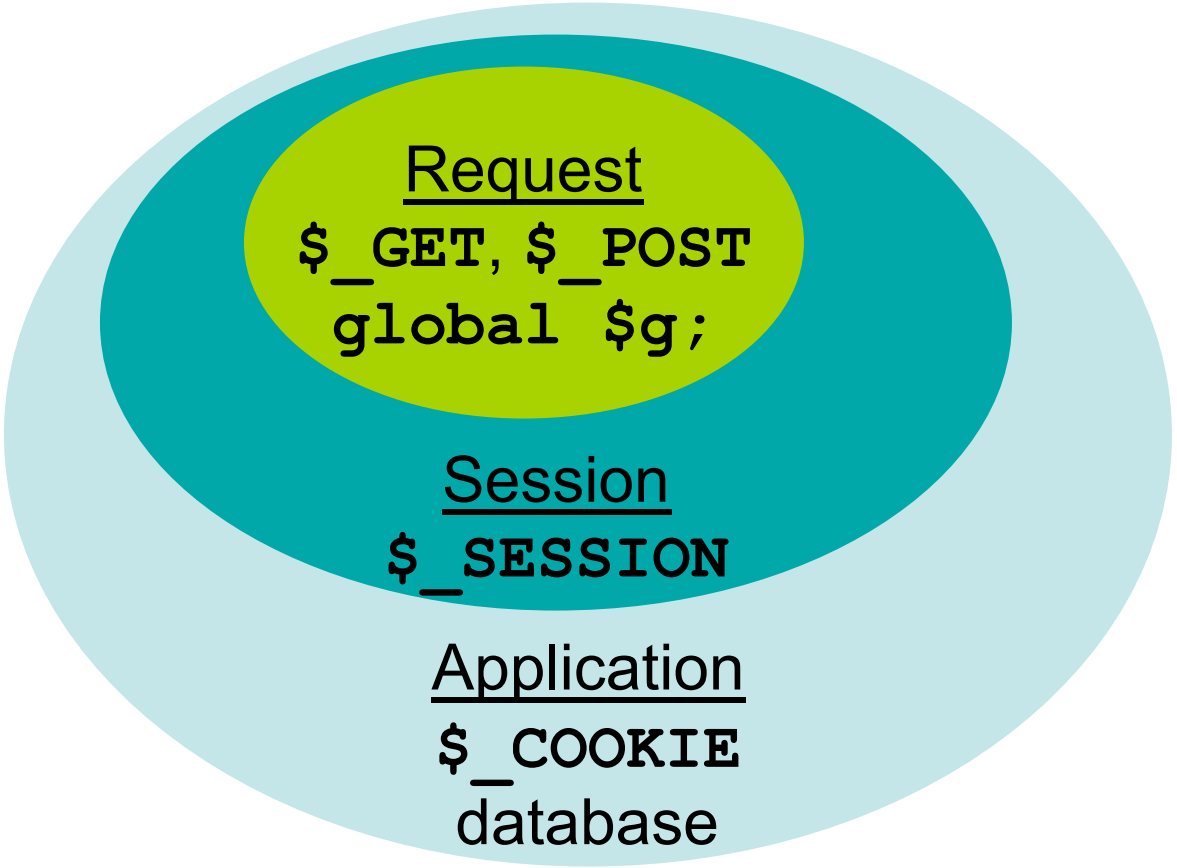
Databases

Outline

- Stateful Web Applications
- AJAX

Concepts

Scope in Server-Side Scripts



Example

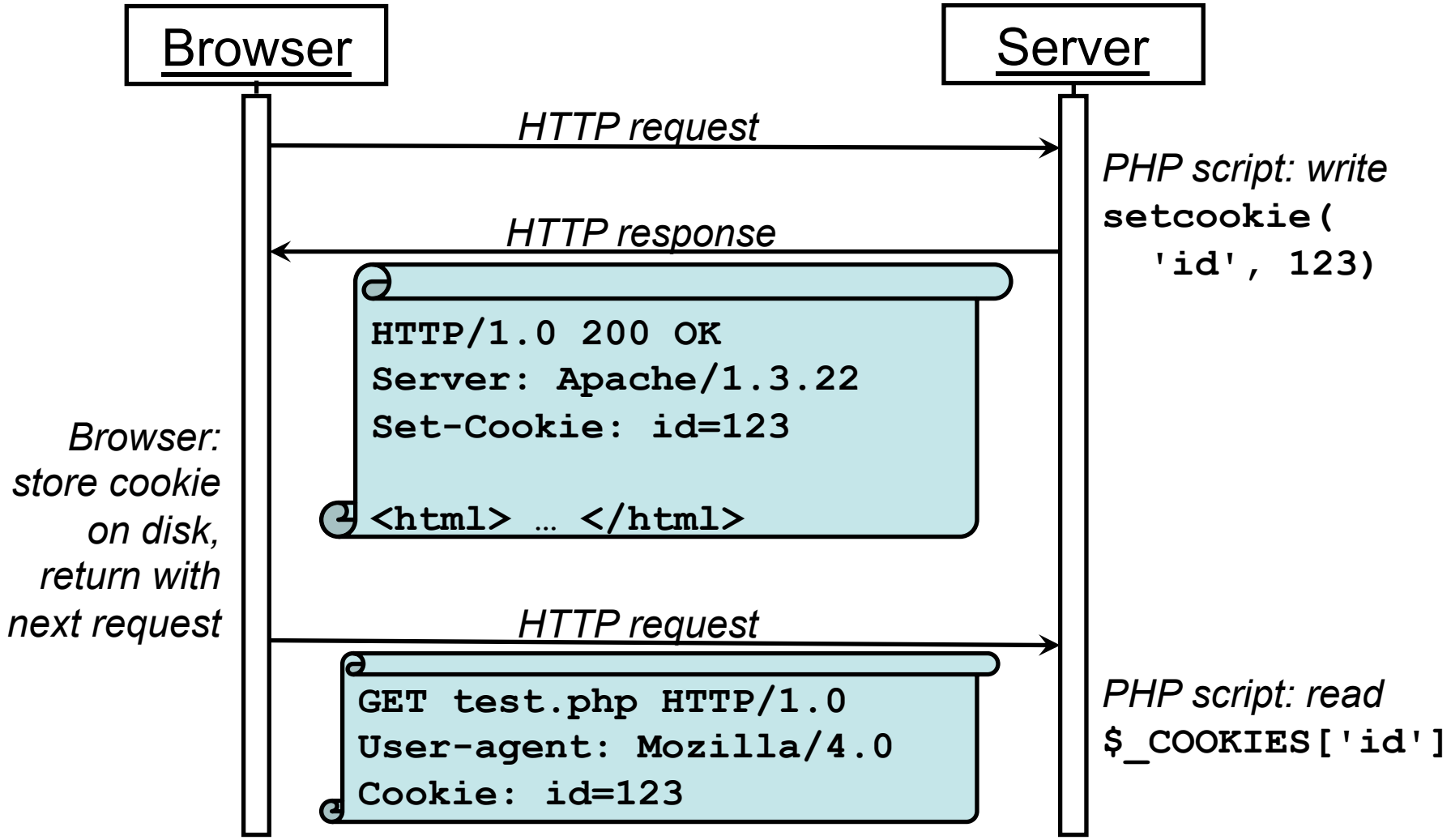
Change <..dir..> to your home directory; create data directory; chmod a+rwx $HOME/data

```
<?php
$d = sqlite_open("<..dir..>/data/sqlite3", 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/sqlite3 to start over<br>";
?>
```

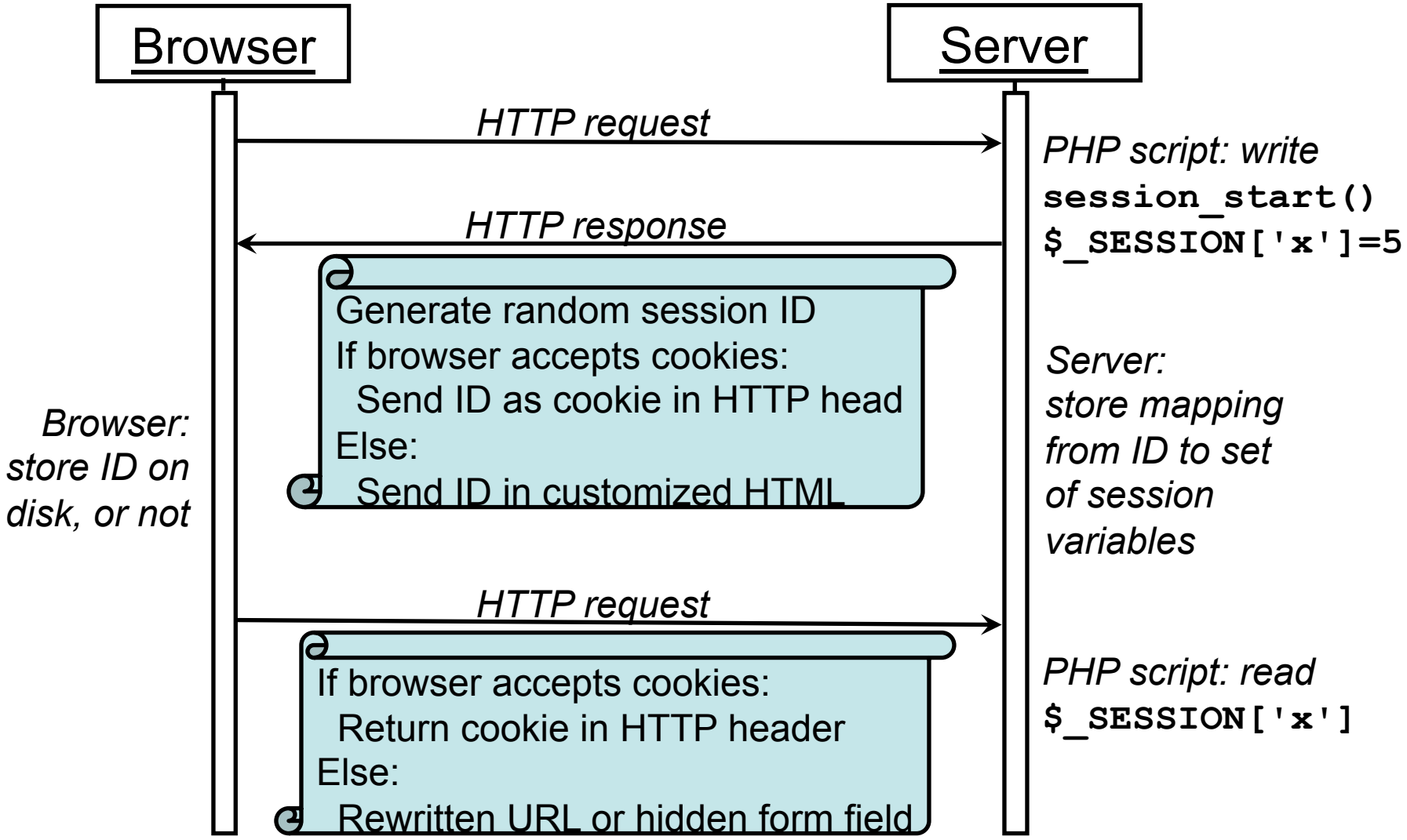
PHP Bindings for Database

- 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

HTTP Cookies



Session Variables



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 server host
Longer lifetime	Deleted after session

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

Outline

- Stateful Web Applications
- AJAX

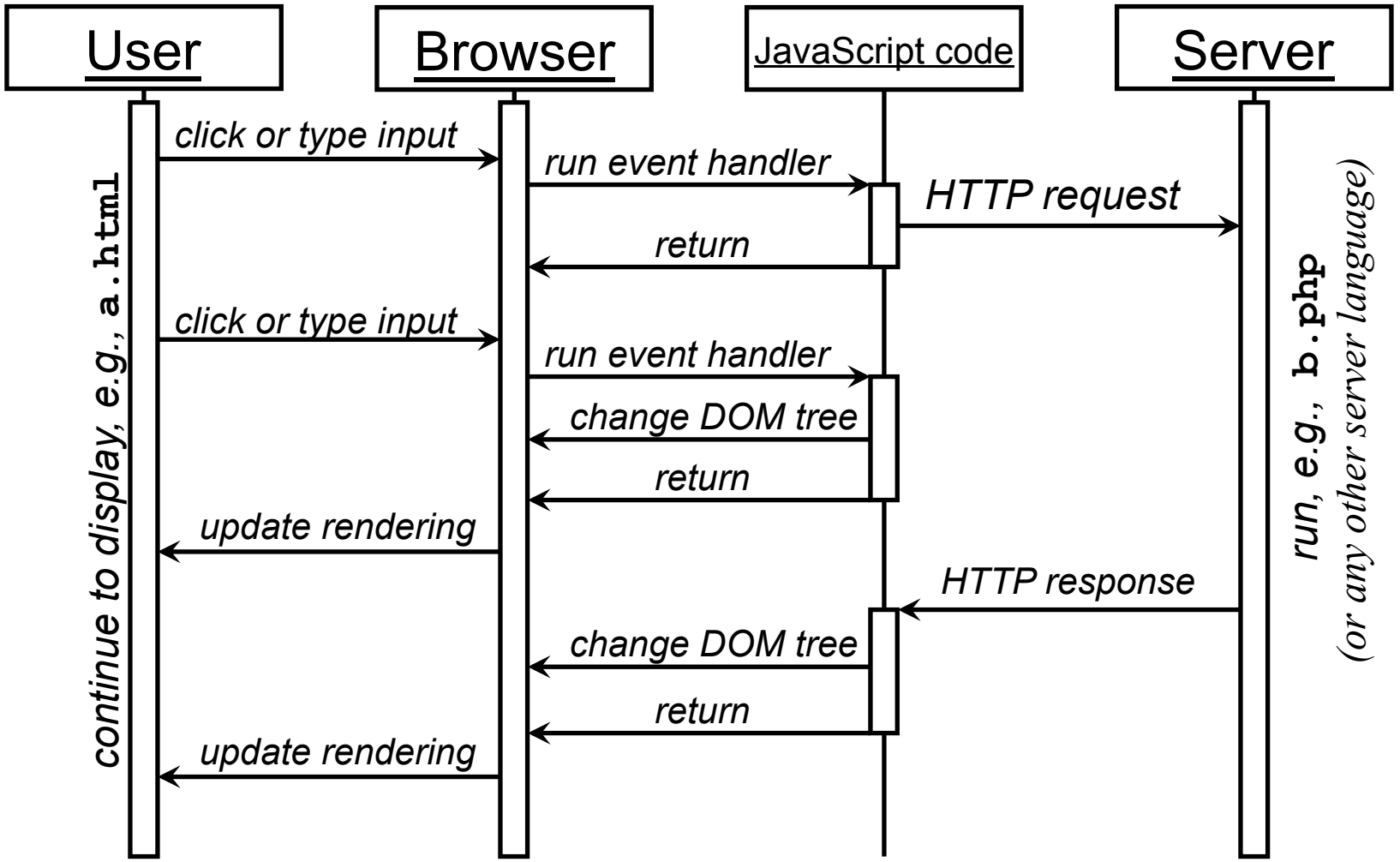
Soap-box

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

Concepts

AJAX Interaction



XMLHttpRequest Objects

Value properties

- **onreadystatechange**
  //*your event handler function*
- **readyState**
  //0 **uninitialized**
  //1 **loading**
  //2 **loaded**
  //3 **interactive**
  //4 **complete**
- **responseText //***string*
- **responseXML //***Document*
- **status //***int: HTTP code*
- **statusText**
  //*string: HTTP reason phrase*

Function properties

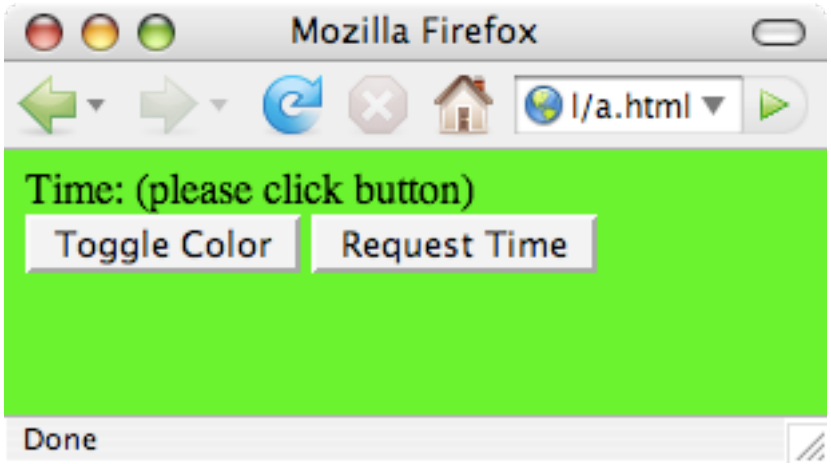
- **abort()**
- **getAllResponseHeaders()**
- **getResponseHeader(**
  parameterName)
- **open(method, url,**
  /*bool***/ asynchronous)**
- **send(content)**
- **setRequestHeader(**
  parameterName,
  parameterValue)

AJAX Example: Files

```
en-cs-cs5142:> pwd
/home/soule/public_html
en-cs-cs5142:> ls -l a.html b.php c.js
-rw-r--r-- 1 soule 1088 387 Oct  2 20:19 a.html
-rw-r--r-- 1 soule 1088  38 Oct  2 20:14 b.php
-rw-r--r-- 1 soule 1088 753 Otc  2 20:19 c.js
en-cs-cs5142:>
```

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>
```



AJAX Example: `b.php`

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

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.'); } } }
  xhr.open('GET', 'b.php?nonce=' + Math.random(), true);
  xhr.onreadystatechange = handleResponse;
  xhr.send(null);
}
```

AJAX Example

```
// Assign handlers, and remember the response
var jqxhr = $.ajax("b.php")
   .done(function() {
      // successful callback return
   })
   .fail(function() {
      // report error
   })
   .always(function() {
      // check textStatus for success or failure
   });

// Do other work here….
```

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

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)

Last Slide

- Next week: quiz2
- Today's lecture
  - Web applications
  - Databases
  - Session state
  - Form validation
  - AJAX
- Next lecture
  - Debugging for scripting languages