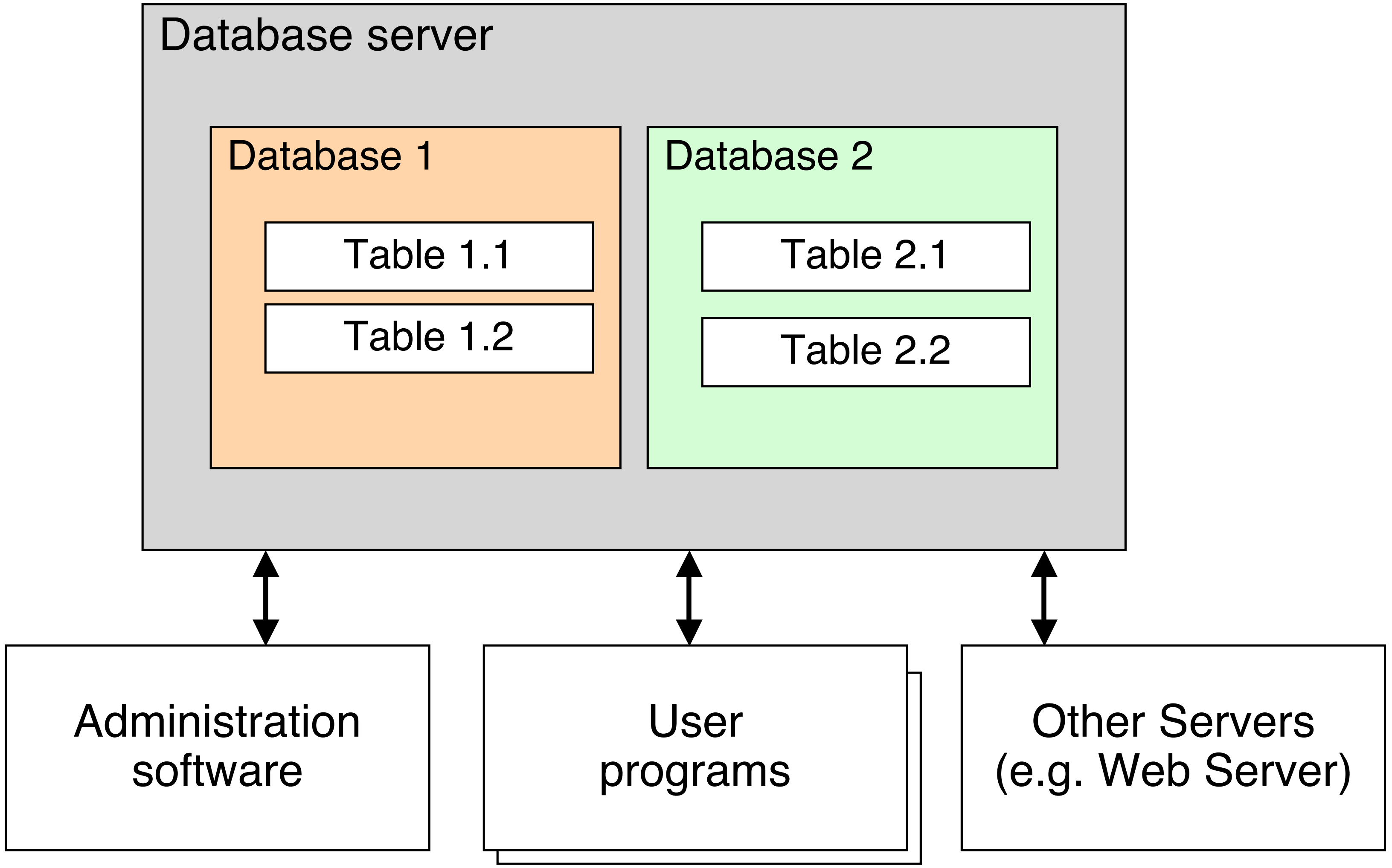
Chapter 3: Interactive Web Applications

Database Management Systems: A Quick Reminder

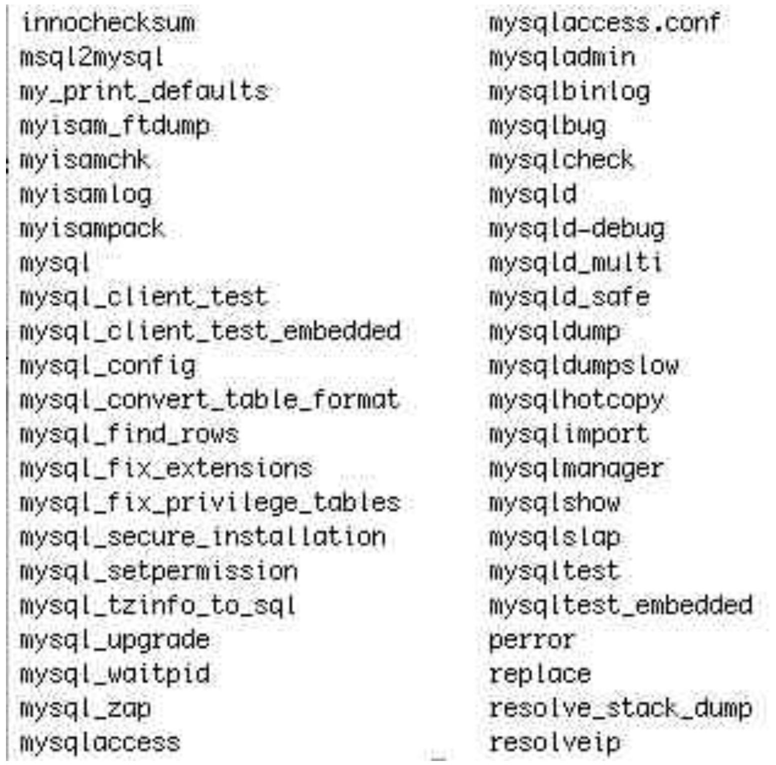
- Database:
  - Structured collection of data items
  - Stored persistently
  - Provides acess to a common data pool for multiple users
- Database Management System (DBMS):
  - Collection of programs for administration and usage of a database
  - Various base models for DBMS:
    - » Old: network model, hierarchical model
    - » Dominant: relational model
    - » Alternative: object-oriented model
- Relational databases:
  - Good methodological support for design of data schema
  - Standardized language interface SQL (Structured Query Language)

Prerequisites and Basic Architecture



MySQL

- Open source software system
  - Frequently used also in commercial context
  - www.mysql.com
- Software package providing:
  - Database server (mysqld)
  - Administration program (mysqladmin)
  - Command line interface (mysql)
  - Various utility programs
- Communication between programs on local host: *socket* interface
  - Bidirectional data stream exchange between programs
  - Similar to files



```
innochecksum                  mysqlaccess.conf
msql2mysql                    mysqladmin
my_print_defaults             mysqlbinlog
myisam_ftdump                 mysqlbug
myisamchk                     mysqlcheck
myisamlog                     mysqld
myisampack                    mysqld-debug
mysql                         mysqld_multi
mysql_client_test             mysqld_safe
mysql_client_test_embedded    mysqldump
mysql_config                  mysqldumpslow
mysql_convert_table_format    mysqlhotcopy
mysql_find_rows               mysqlimport
mysql_fix_extensions          mysqlmanager
mysql_fix_privilege_tables    mysqlshow
mysql_secure_installation     mysqlslap
mysql_setpermission           mysqltest
mysql_tzinfo_to_sql           mysqltest_embedded
mysql_upgrade                 perror
mysql_waitpid                 replace
mysql_zap                     resolve_stack_dump
mysqlaccess                   resolveip
```

Before Creating Anything in the Database...

- Using a database requires careful *information design.*
- Which are the data to be stored?
- Are there existing data to connect to?
- What is the ***schema*** of the data to be stored?
  - Eg. Entity-Relationship diagrams as a tool
  - Transformation into relational database schema (table design)
- Once a database if filled with data and in use, it is rather difficult to modify!
  - Database schema design has to be carried out with great care!
- Most important rule: Avoid redundant storage of information

Creating Database Tables (1)

- Prerequisites:
  - Database server running
  - Socket connection between programs intact
  - User accounts with adequate privileges known
- First step: Create ***database***
  - Container for many tables
  - Requires special privileges
  - Example SQL:

    ```
    create database music;
    ```
- Second step: ***Select*** database
  - Sets the context for further interactions
  - Example SQL:

    ```
    use music
    ```

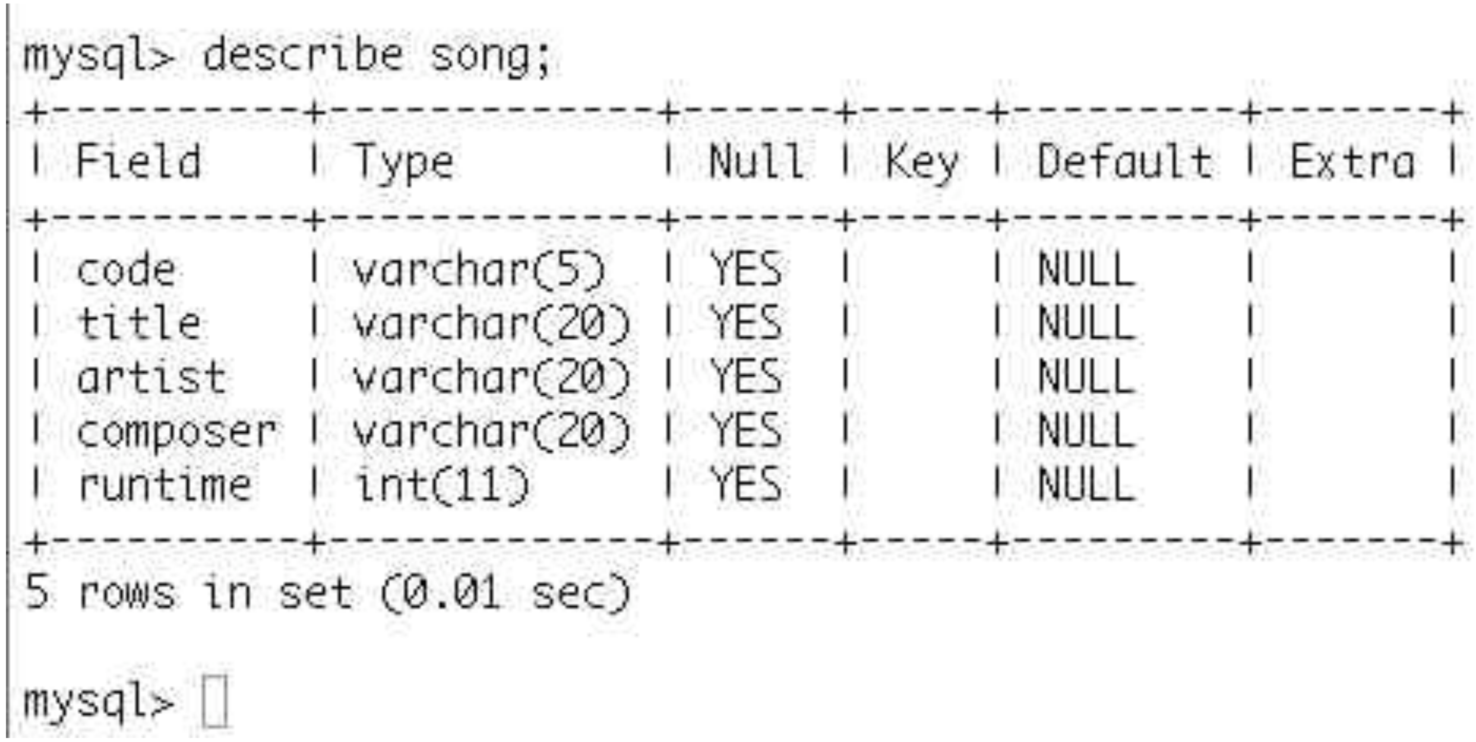
Creating Database Tables (2)

- Third step: Create ***tables***
  - According to earlier design
  - Each table should provide a unique identifier ***(primary key)***
  - SQL Example:

    ```
    create table song (code VARCHAR(5), title VARCHAR(20),
    artist VARCHAR(20), composer VARCHAR(20), runtime INT);
    ```
  - Further steps: Defining keys, indices etc.
- Fourth step: Fill tables with ***data***
  - Simplest case: Individual SQL commands
  - Better: Import from structured data file
  - Frequent: Special programs for importing and creating data
  - SQL Example:

    ```
    insert into song
    values ('1','One','U2','Adam Clayton, Bono, Larry Mullen
    & The Edge',272);
    ```

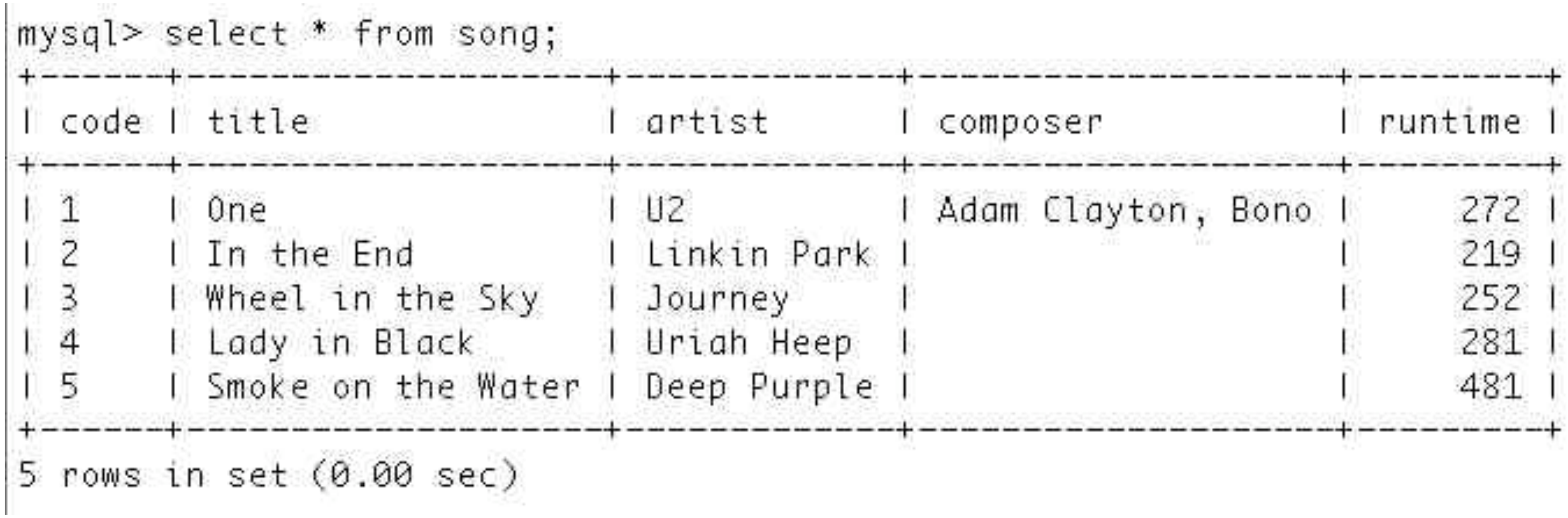
SQL Monitor Output



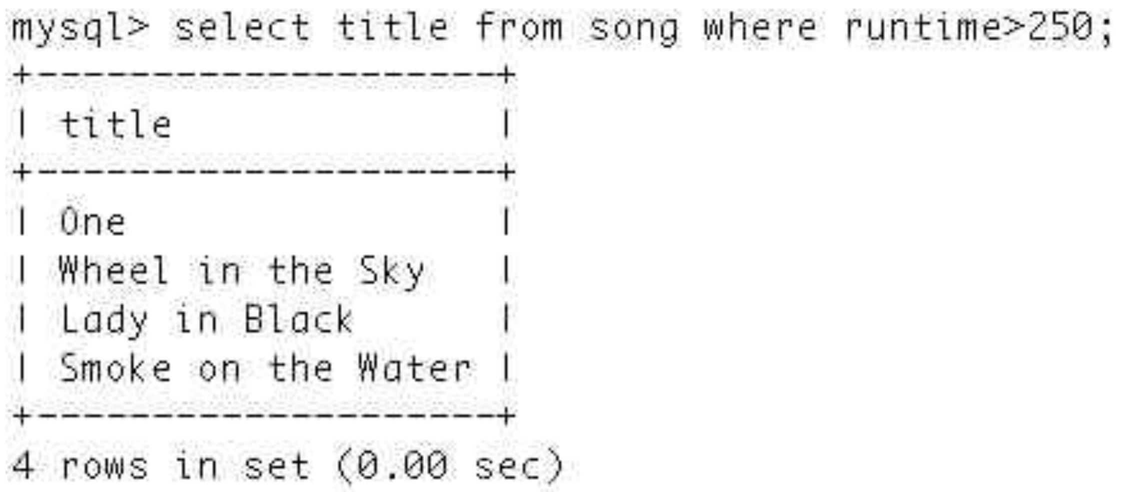
```
mysql> describe song;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| code     | varchar(5)  | YES  |     | NULL    |       |
| title    | varchar(20) | YES  |     | NULL    |       |
| artist   | varchar(20) | YES  |     | NULL    |       |
| composer | varchar(20) | YES  |     | NULL    |       |
| runtime  | int(11)     | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)

mysql> 
```

Queries with SQL



```
mysql> select * from song;
+------+--------------------+-------------+---------------------+---------+
| code | title              | artist      | composer            | runtime |
+------+--------------------+-------------+---------------------+---------+
| 1    | One                | U2          | Adam Clayton, Bono  |     272 |
| 2    | In the End         | Linkin Park |                     |     219 |
| 3    | Wheel in the Sky   | Journey     |                     |     252 |
| 4    | Lady in Black      | Uriah Heep  |                     |     281 |
| 5    | Smoke on the Water | Deep Purple |                     |     481 |
+------+--------------------+-------------+---------------------+---------+
5 rows in set (0.00 sec)
```



```
mysql> select title from song where runtime>250;
+--------------------+
| title              |
+--------------------+
| One                |
| Wheel in the Sky   |
| Lady in Black      |
| Smoke on the Water |
+--------------------+
4 rows in set (0.00 sec)
```

Databases, PHP and MySQL

- Special libraries for database access:
  - "Database extensions"
  - Generic for all database systems
- For specific database systems:
  - "Vendor specific database extensions"
- For MySQL:
  - MySQL-specific database extensions to PHP

Connecting to a Database from PHP

- First step: ***Connect*** to server
  - Establish a connection for data exchange between Web Server/PHP plugin and database server
  - Often local (sockets), if both programs on same machine
  - Requires hostname, (database) username, password
  - PHP function: **`mysql_connect()`**
    - » Returns a link (resource) which can be used for **`mysql_close()`**
- Second step: ***Select*** a database
  - Corresponds to the SQL command **`use`**
  - Requires database name (and possibly link to server)
  - PHP function: **`mysql_select_db()`**
    - » Returns Boolean result (success)

Example: Connecting to Database

```
<?php

$link = mysql_connect('localhost','root','demopw')
  or die ('Could not connect: '.mysql_error());
echo 'Connected.<br/>';

mysql_select_db('music')
  or die ('Could not select db.');
echo 'DB selected.<br/>';


...
?>
```

Sending Database Queries from PHP

- Basic idea (as in all programming language/database integrations):
  - SQL queries are given as strings to library functions
- Most important function in MySQL extensions to PHP:
  `mysql_query()`
  - Requires SQL query as parameter (optionally link to server as 2nd param.)
  - "Query" includes also **`INSERT`**, **`UPDATE`**, **`DELETE`**, **`DROP`** (SQL)!
- Return value in case of **`SELECT`**, **`SHOW`**, **`DESCRIBE`** and similar:
  - Result set represented by resource value
  - Special functions to retrieve result data as PHP data structures
  - **`mysql_num_rows()`**
    - » Number of rows returned
  - **`mysql_fetch_array()`**
    - » Reads one row of data and transforms it into an array
    - » Makes the next row available

Example: Reading Data From a Query in PHP

```
<?php
...
$query = 'SELECT * FROM song';
$result = mysql_query($query);

while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
   foreach ($row as $element) {
      echo $element;
      echo ', ';
   }
   echo("<br/>");
...
?>
```

dbaccess.php

Creating HTML Output From SQL Query (1)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
  Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
  <title>Database table in HTML</title>
</head>

<?php
$link = mysql_connect('localhost','root','demopw')
 or die ('Could not connect: '.mysql_error());
mysql_select_db('music') or die ('Could not select db.');
?>
```

dbaccess_html.php

Creating HTML Output From SQL Query (2)

```
...
<body>
  <h1>The following table is retrieved from MySQL:</h1>
  <table>
      <?php
      $query = 'SELECT * FROM song';
      $result = mysql_query($query)
            or die ('Query failed'.mysql_error());
      while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
            echo "\t<tr>\n";
            foreach ($row as $element) {
                  echo "\t\t<td>";
                  echo $element;
                  echo "</td>\n";
            }
            echo "\t</tr>\n";
      }
      ?>
  </table>
```

Creating HTML Output From SQL Query (3)

```
...
<?php
  mysql_free_result($result);
  mysql_close($link);
?>

</body>
</html>
```

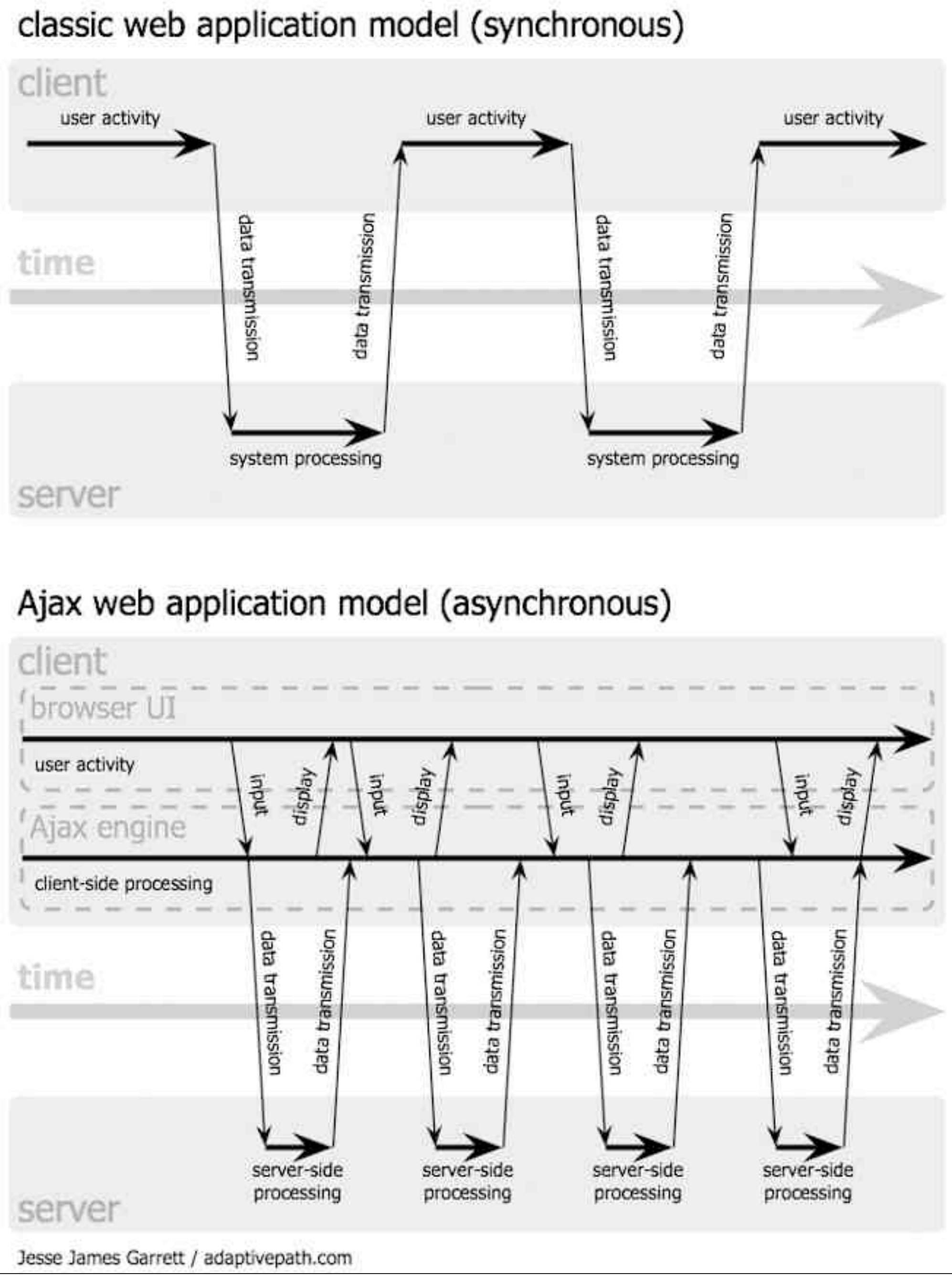
Chapter 3: Interactive Web Applications

Christian Wenz: Ajax - schnell und kompakt. entwickler.press 2007

Asynchronous JavaScript + HTML (AJAX)

- James Garrett 2005:
  http://www.adaptivepath.com/ideas/essays/archives/000385.php
- Catchy name for an idea which was in use already at the time:
  - Google Suggest
  - Google Maps
- Basic idea:
  - Loading data from server is decoupled from changes in the presentation
- Advantages:
  - User can interact fluidly with the application
  - Information from server is fetched at regular intervals - display can always stay up-to-date
- AJAX is not a technology, it is a combination of known technologies
  - XHTML, CSS, DOM, XML, XSLT, JavaScript, XMLHttpRequest
- There are AJAX-like applications which use neither JavaScript nor HTML
  - E.g. using Flash and querying servers in the background

Asynchronicity



Examples:

Web mail access

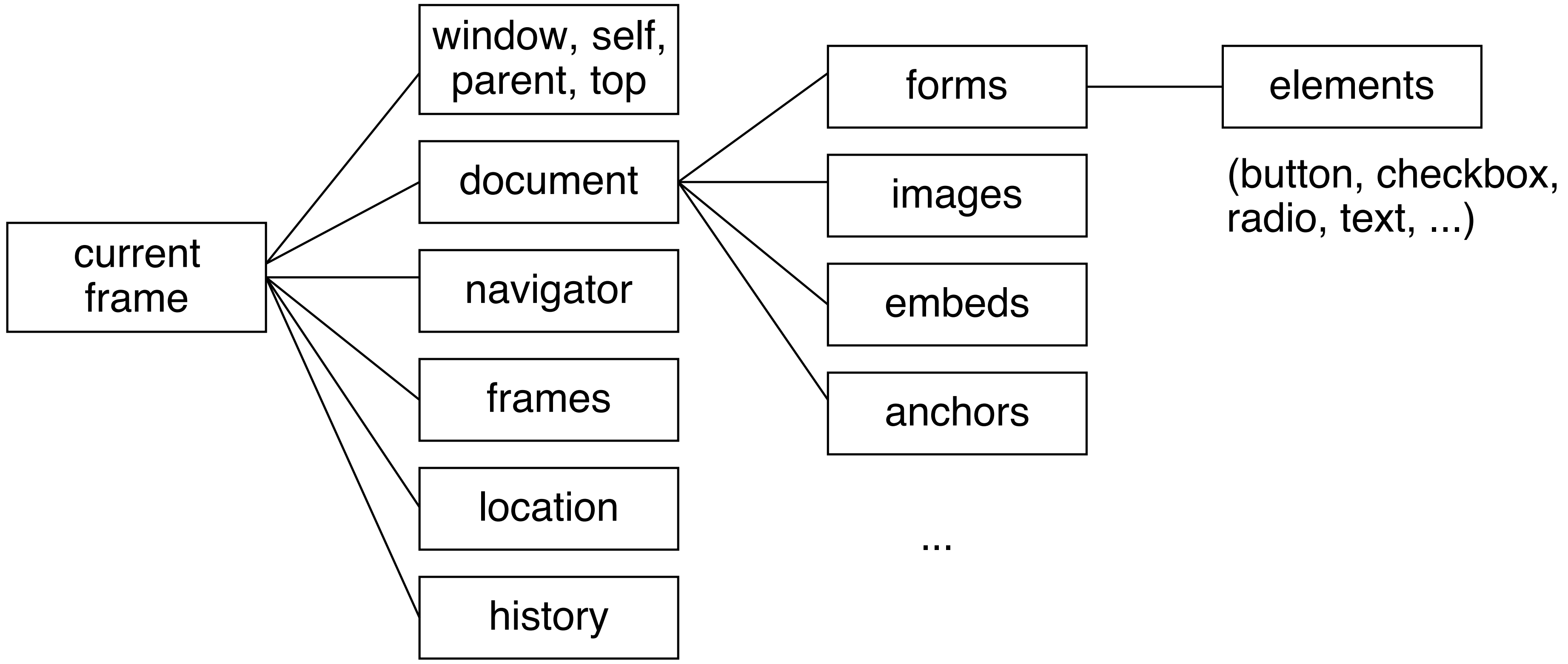
Autocompletion of forms (e.g. City based on Zip code)

AJAX and Client-Side Scripting

- AJAX applications are programs executed in the Web browser
  - Require a runtime environment
  - Usually programmed in JavaScript
- AJAX applications need to modify or construct HTML to be displayed in the browser
  - Requires acess to loaded/displayed HTML
  - *Domain Object Model* (DOM) is used for accessing and manipulating page content

JavaScript Object Tree

- Elements of the displayed document and other information can be accessed and manipulated
- Navigation:
  - Mostly selection by "id"
  - Starting point is often "document" object



DOM Reminder

- DOM is a collection of functions which make it possible to access and manipulate HTML and XML documents in the browser
- DOM ist a standardised API (Application Programming Interface)
  - Usable with several programming languages
- Examples of DOM object properties and methods:

```
nodeName, nodeValue, nodeType, attributes
getElementbyId()
parentNode, hasChildNodes();
childNodes, firstChild, lastChild, previousSibling,
nextSibling;
createElement(); createTextNode();
insertBefore(), replaceChild(), removeChild(),
appendChild();
```

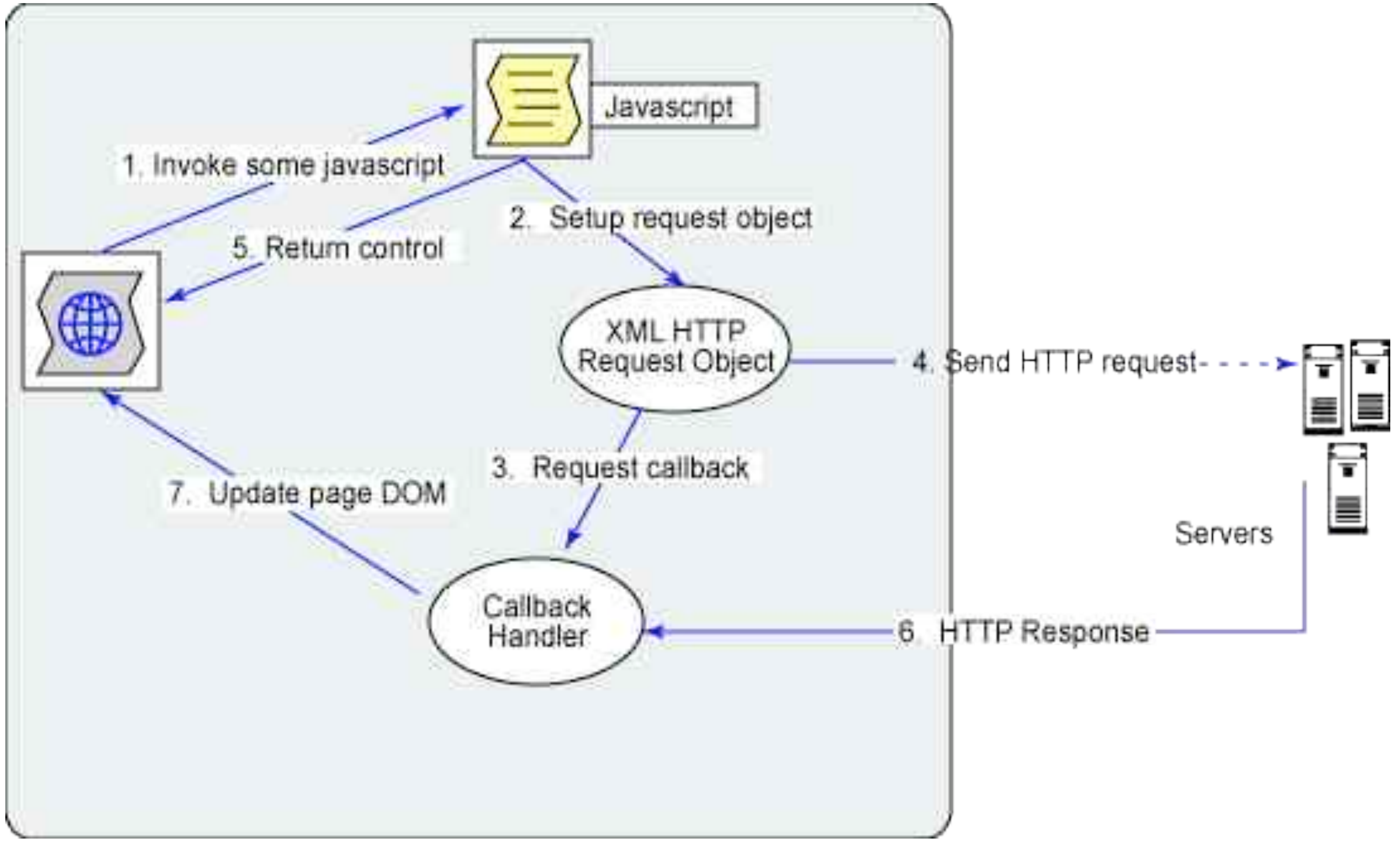
AJAX and Server-Side Scripting

- AJAX applications make particular sense when the data loaded from the server changes dynamically
  - PHP scripts or other server-side dynamics
  - Database connectivity
- For ease of understanding (only!):
  - Most examples in the following deal with static Web pages only

Request Construction and Handling

- Main functionalities required:
  - Construction of a request to be sent to the server
  - Sending a request to the server
  - Waiting (asynchronously) until server responds
  - Calling functions to analyze server response
- All these functionalities are realized in one single object (in the sense of object-orientation):
  - XMLHttpRequest

Basic Control Flow



http://www.ibm.com/developerworks, Dojo framework

XMLHttpRequest

- Outlook Web Access for Internet Explorer 5 (end 90s):
  - XMLHttpRequest object invented at Microsoft
  - Realized as ActiveX object
- Mozilla 1.4 (Netscape 7.1) and derivates (including Firefox):
  - Native XMLHttpRequest object for JavaScript
  - Independent of Active X
- Other manufacturers:
  - Followed step by step: Konqueror, Apple Safari, Opera, iCab
- Since Internet Explorer 7 ActiveX no longer required
  - Just JavaScript
- Under W3C standardization (Working draft August 2009)
- Long term situation for creating XMLHttpRequest object will be:

```
var XMLHTTP = new XMLHttpRequest();
```

- Currently we have to fight with browser incompatibilities!
  - Frameworks like *Prototype* can help

Platform Independent Creation of XMLHttpRequest

```
var XMLHTTP = null;
if (window.XMLHttpRequest) {
    XMLHTTP = new XMLHttpRequest();
} else if (window.ActiveXObject) {
    try {
        XMLHTTP = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (ex) {
        try {
            XMLHTTP = new ActiveXObject("Microsoft.XMLHTTP");
        } catch (ex) {}
    }
}
```

IE < 7.0 or Standard

For older IE versions than 6.0

Construction of a HTTP Request

- `open()` method of XMLHTTPRequest object
  - Note: There is no interaction with the server involved, despite the name
- Required parameters:
  - HTTP method: GET, POST or HEAD
  - URL to send the request to
- Optional parameters:
  - Asynchronous or synchronous treatment (default asynchronous = true)
  - Username and password for authentication

- Examples:

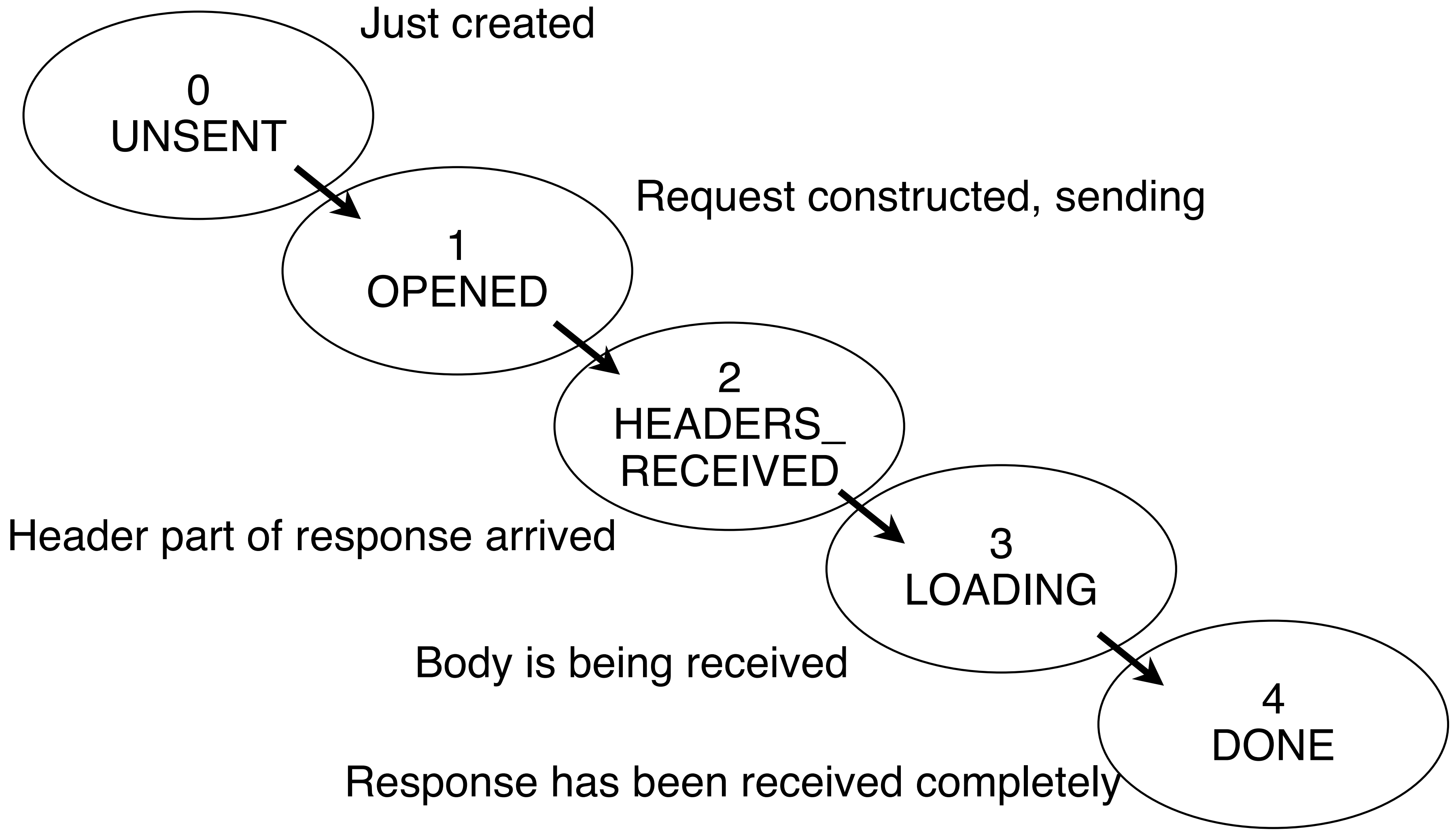
```
XMLHTTP.open("GET", "fibonacci.php?fib=12")
XMLHTTP.open("POST", "/start.html", false, un, pwd);
```

Sending a Request

- Before sending: **`XMLHTTP.setRequestHeader()`**
  - Setting headers for the request
  - Recommended: **`Content-Type`** (MIME type)

- **`XMLHTTP.send()`**
  - Sends request to server
- Parameter:
  - In the simplest case (in particular GET method): **`null`**
  - For more complex cases:
    "Request entity body" is given as parameter
    - » Mainly for POST method

States of an XMLHttpRequest Object


Just created
0
UNSENT
1
OPENED
Request constructed, sending
2
HEADERS_
RECEIVED
Header part of response arrived
3
LOADING
Body is being received
4
DONE
Response has been received completely

Asynchronous Reaction by Event Handler

- In order to react to the received response:
  - Function has to be called when state 4 is reached
- Registering an event handler:
  - Callback function, called when event takes place
  - Similar to event handling for user interfaces (Java Swing, Flash)
- For Ajax:
  - Callback method registered with XMLHTTPRequest object
  - Event = State 4 is reached
  - More general: Called at any state change
- **`XMLHTTP.onreadystatechange =`** ***`function;`***
  - **`XMLHTTP.readystate`** gives current state (as number)

Example: Very Simple Request

```
...<body>
<script type = "text/javascript">
  var XMLHTTP = new XMLHttpRequest();

  function dataOutput() {
    if (XMLHTTP.readyState == 4) {
      var d = document.getElementById("data");
      d.innerHTML += XMLHTTP.responseText;
    }
  }

  window.onload = function() {
      XMLHTTP.open("GET", "data.txt", true);
      XMLHTTP.onreadystatechange = dataOutput;
      XMLHTTP.send(null);
  }
</script>

</body>
  <p id="data">Data from server: </p>
</html>
```

AJAX and XML

- The server response (essentially text) needs to be analysed
- XML
  - Supports arbitrarily structured information
  - Is fully supported by JavaScript and DOM
- Servers should return data as XML
- Problem (currently):
  - Browser incompatibilities

Example XML Data

```
<?xml version="1.0" encoding="UTF-8"?>
<ResultSet totalResultsAvailable="24900000"
   totalResultsReturned="10">
  <Result>
    <Title>AJAX - Wikipedia</Title>
    <Summary>Background about the web development technique for
   creating interactive web applications.</Summary>
    <Url>http://en.wikipedia.org/wiki/AJAX</Url>
  </Result>
  <Result>
    <Title>Ajax: A New Approach to Web Applications</Title>
    <Summary>Essay by Jesse James Garrett from Adaptive
   Path.</Summary>
    <Url>http://www.adaptivepath.com/p...s/000385.php</Url>
  </Result>
  <Result>
    <Title>AFC Ajax</Title>
    <Summary>Official site. Club information, match reports, news,
   and much more.</Summary>
    <Url>http://www.ajax.nl/</Url>
  </Result>
</ResultSet>
```

From C.Wenz

AJAX Program Creating a HTML Table from XML

- Fixed HTML text :

```
<body>
  <p>
    <span id="Anzahl">0</span> von
    <span id="Gesamt">0</span> Treffern:
  </p>

  <table id="Treffer">
    <thead>
     <tr><th>Titel</th><th>Beschreibung</th><th>URL</th></tr>
    </thead>
  </table>
</body>
```

Script has to fill the missing data from XML response.
Basic structure of script as above.

From C.Wenz

Transformer Callback Function (1)

```
function DatenAusgeben() {
  if (XMLHTTP.readyState == 4) {
    var xml = XMLHTTP.responseXML;

    var anzahl = document.getElementById("Anzahl");
    var gesamt = document.getElementById("Gesamt");
    anzahl.innerHTML =
  xml.documentElement.getAttribute("totalResultsReturned");
    gesamt.innerHTML =
  xml.documentElement.getAttribute("totalResultsAvailable");

    var treffer = document.getElementById("Treffer");
    var tbody = document.createElement("tbody");

    var ergebnisse = xml.getElementsByTagName("Result");
    ...
```

Transformer Callback Function (2)

```
... for (var i=0; i<ergebnisse.length; i++) {
      var zeile = document.createElement("tr");
      var titel = document.createElement("td");
      var beschreibung = document.createElement("td");
      var url = document.createElement("td");
      var titeltext, beschreibungtext, urltext;
      for (var j=0; j<ergebnisse[i].childNodes.length; j++) {
        var knoten = ergebnisse[i].childNodes[j];
        switch (knoten.nodeName) {
          case "Title":
            titeltext = document.createTextNode(
              knoten.firstChild.nodeValue);
            break;
          case "Summary":
            beschreibungtext = document.createTextNode(
              knoten.firstChild.nodeValue);
            break;
          case "Url":
            urltext = document.createTextNode(
              knoten.firstChild.nodeValue);
            break;
        }
      }
```

Transformer Callback Function (2)

```
... for (var i=0; i<ergebnisse.length; i++) {
       ...
       for (var j=0; j<ergebnisse[i].childNodes.length; j++) {
       ...
       titel.appendChild(titeltext);
       beschreibung.appendChild(beschreibungtext);
       url.appendChild(urltext);

       zeile.appendChild(titel);
       zeile.appendChild(beschreibung);
       zeile.appendChild(url);
       tbody.appendChild(zeile);
     }
     treffer.appendChild(tbody);
  }
}
```

AJAJ? – Simple Serialization with JSON

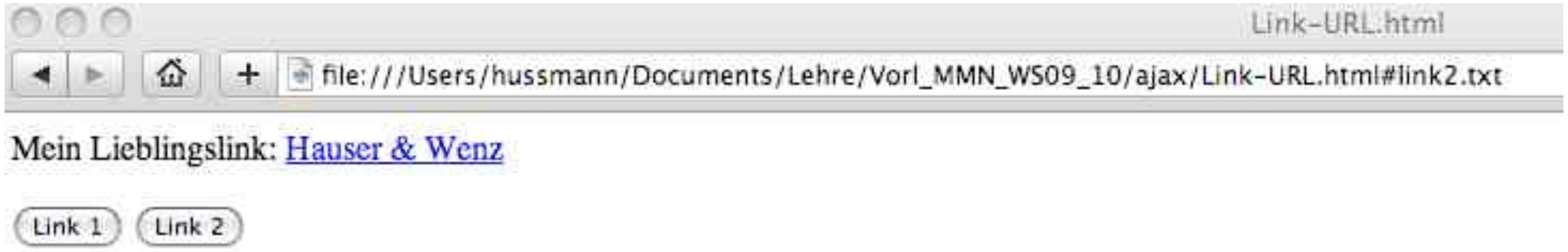
- XML Serialization of data
  - Tends to be long
  - Many redundant elements
  - Occupies a lot of bandwidth
- Alternative Serialization: JSON (JavaScript Object Notation)

```
{
"ResultSet":
  {
  "totalResultsAvailable": "24900000",
  "totalResultsReturned": 10,
  "Result":
    [
      {
      "Title": "AJAX - Wikipedia",
      "Url":"http:\/\/en.wikipedia.org\/wiki\/AJAX"
      },
      {
      "Title":"Ajax: A New Approach to Web Applications",
      "Url":"http:\/\/www.adaptivepath.com\/p.../000385.php"
      }
    ]
  }
}
```

Problems with AJAX

- Back button
  - Browsers do not store dynamically modified pages in history
- Polling
  - Browser send more requests at a more regular pace; i.e the base assumptions for traffic engineering change
- Bookmarks
  - It is difficult to set a bookmark at a specifc state of a dynamically created flow of pages
  - Solution attempts use the document-internal anchors (#)
- Indexing by search engines

Example: Bookmarking Support


Link-URL.html
file:///Users/hussmann/Documents/Lehre/Vorl_MMN_WS09_10/ajax/Link-URL.html#link2.txt
Link 1
Link 2

Mein Lieblingslink: Hauser & Wenz

When processing response:

```
location.hash = "#" + escape(url);
```

```
window.onload = function() {
  if (location.hash.length > 1) {
    url = unescape(location.hash.substring(1));
    ladeURL();
  }
}
```

Sajax: Framework for AJAX (in PHP)

- Example for a framework supporting Ajax
- Sajax (Simple Ajax)
  - http://www.modernmethod.com/sajax
  - Open Source
  - Framework (library) for several scripting languages, including PHP
- Abstracts from technical details of AJAX
  - Write AJAX applications without knowing about XMLHttpRequest
- Basic idea:
  - Create a server-side dynamic function (in PHP)
  - "Export" this function with Sajax (**`sajax_export('`*`functionname`*`')`**)
  - In the JavaScript section of the page, call **`sajax_show_javascript()`** (a PHP function generating JavaScript)
  - Corresponding to the server-side function, now a JavaScript function exists (**`x_`*`functionname`***) which calls the server-side function asynchrounously (i.e. a callback function is given as parameter)

Examples of AJAX Applications

- Maps: Google Maps, OpenStreetMap
- Office: AjaxWrite, nexImage
- Social software: Flickr, Del.icio.us, Last.fm, *VZ Netzwerke
- Mail: Google Mail
- Web search: Google Suggest
- CRM: 24SevenOffice