Chapter 2: Interactive Web Applications

2.1 Interactivity and Multimedia in the WWW architecture

2.2 Client-Side Multimedia in the Web
   (Example HTML5)

2.3 Interactive Server-Side Scripting (Example PHP)

2.4 Data Storage in Web Applications
   (Example Database Access in PHP)

2.5 Integrated Server/Client-Side Scripting
   (Example jQuery/AJAX)
Database Management Systems: A Quick Reminder

• Database:
  – Structured collection of data items
  – Stored persistently
  – Provides access 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)

• Document-oriented databases:
  – Based on document trees, APIs for queries (“NoSQL”)
Warm-Up (Data Base Technology)

- Who knows brand names of commonly used relational database systems?
- Who has actually written program code using one of them?
- What is SQL?
- Who knows about alternative approaches to databases different from relational ones?
- Were there other historic database systems before the invention of relational databases?
Prerequisites and Basic Architecture

Database server

Database 1
- Table 1.1
- Table 1.2

Database 2
- Table 2.1
- Table 2.2

Administration software

User programs

Other Servers (e.g. Web Server)
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
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?
  - E.g. Entity-Relationship diagrams as a tool
  - Transformation into relational database schema (table design)
- Once a database is filled with data and in use, it is difficult to modify!
  - Database schema design has to be carried out with great care!
- Most important rule: Avoid redundant storage of information
  - But keep performance in mind...
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:
    ```sql
    create database music;
    ```

- Second step: **Choose used** database
  - Sets the context for further interactions
  - Example SQL:
    ```sql
    use music
    ```
Do Not Get Confused!

- Why are some commands to the database monitor terminated with a semicolon, and some are not?
- Isn't it terribly dangerous that I show you the (weak) password for 'root'?  
  – Can everybody now break into my computer?
Creating Database Tables (2)

- Third step: Create **tables**
  - According to earlier design
  - Each table should provide a unique identifier (**primary key**)
  - SQL Example:
    ```sql
    create table mysongs (code VARCHAR(5) primary key,
                          title VARCHAR(20), artist VARCHAR(20),
                          album VARCHAR(20), runtime INT);
    ```

- 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:
    ```sql
    insert into mysongs
    values ('1','One','U2','The Complete U2',272);
    ```
### SQL Monitor Output

```
mysql> describe mysongs;

+-------------------+---------------+------+-----+---------+----------------+
| Field             | Type          | Null | Key | Default | Extra          |
+-------------------+---------------+------|-----|--------|----------------|
| code              | varchar(5)    | NO   | PRI | NULL   |                |
| title             | varchar(20)   | YES  |     | NULL   |                |
| artist            | varchar(20)   | YES  |     | NULL   |                |
| album             | varchar(20)   | YES  |     | NULL   |                |
| runtime           | int(11)       | YES  |     | NULL   |                |
+-------------------+---------------+------|-----|--------|----------------|

5 rows in set (0.01 sec)
```
**Queries with SQL**

```sql
mysql> select * from mysongs;
+------------+---------+---------+-------------+----------+
| code | title      | artist  | album         | runtime |
+------------+---------+---------+-------------+----------+
| 1     | One       | U2      | The Complete U2 | 272     |
| 2     | In the End | Linkin Park | Hybrid Theory | 216     |
| 3     | Wheel in the Sky | Journey   | Infinity      | 252     |
| 4     | Lady in Black | Uriah Heep | Lady in Black | 281     |
| 5     | Smoke on the Water | Deep Purple | Machine Head | 378     |
| 6     | Analog Man | Joe Walsh | Analog Man    | 243     |
+------------+---------+---------+-------------+----------+
6 rows in set (0.00 sec)
```

```sql
mysql> select title from mysongs where runtime>250;
+---------+
| title   |
+---------+
| One     |
| Wheel in the Sky |
| Lady in Black |
| Smoke on the Water |
+---------+
4 rows in set (0.00 sec)
```
Server-Side Databases, PHP and MySQL

• Libraries for database access:
  – "Database extensions” for server-side scripts
• For PHP and MySQL:
  – MySQL database extensions for PHP usually pre-installed
  – Different APIs for PHP
    » MySQL Improved Extension (mysqli) — used here
    » PHP Data Objects (PDO) interface
  – mysqli allows object-oriented programming
Assume You Are The Software Architect

• Problem to be solved: Access to MySQL data bases from PHP scripts
• How would you design the interface (API)?
• What are desired properties for a well-designed solution?
• What are key obstacles for finding a good design?
Detour (1): Object-Orientation in PHP

- Object-orientation:
  - everything is an object
  - objects have attributes and methods

- In PHP:

  class Song {
    var $title;
    var $artist;
    var $favorite = False;

    public function __construct($title, $artist) {
        $this->title = $title;
        $this->artist = $artist;
    }

    public function like() {
        $this->favorite = True;
    }
  }

  song.php
Detour (2): Using Classes in PHP

• To use the class in another script, it needs to be imported:
  – `require("lecture.php");`
  – `require_once("lecture.php");`

• Afterwards, classes can be **instantiated** and become objects that we can assign to variables:

```
$song1 = new Song();
$song2 = new Song("One", "U2");
```

• Accessing member variables and methods is done using the **arrow notation**:

```
echo 'Title: ' . $song1->title;
$song2->like();
$song2->show();
```
Connecting to a Database from PHP

• Steps:
  – Connect to server
  – Select (use) a database
  – MySQL Improved PHP API: Combined into one step

• Connection as object:
  – Establish a connection for data exchange between Web Server/PHP plugin and database server
  – PHP: Create a new mysqli object
    » Returns an object which can be used for further operations
  – Requires hostname, (MySQL) username, password, database name

• Realization:
  – Local communication (through socket), if both programs on same machine
  – TCP/IP connection to remote server is possible

• Performance optimization:
  – Persistent connections and connection pools
Example: Connecting to Database

```php
$db = new mysqli('localhost', 'root', 'demopw', 'music');

if ($db->connect_error) {
    die('Failed to connect: '.$db->connect_error);
}

echo 'Connected to server and DB selected.<br/>';
...
?>
```
Sending Database Queries from PHP

• Basic idea (in all programming language/database integrations):
  – SQL query given as string to library functions
• MySQL/PHP:
  `query()` method of `mysqli` object
  – Parameter: SQL query (optionally link to server as 2nd param.)
• Return value:
  – `mysqli_result` object
  – Special functions and variables to process result data (examples):
    – `$num_rows` (property)
      » Number of rows in the result set
    – `fetch_assoc()`
      » Reads one row of result data and returns it as associative array
      » Makes the next row available
Example: Reading Data From a Query in PHP

```php
<?php   ... $db = ... connecting, selecting ...
$query = 'SELECT * FROM mysongs';
$result = $db->query($query);
if (!$result) {
    die('Query failed: '. $db->error);
}
while ($row = $result->fetch_assoc()){
    foreach ($row as $element) {
        echo $element;
        echo ', ';
    }
    echo("<br/>");
}
...?

dbaccess.php
```
Creating HTML Output From SQL Query (1)

<!DOCTYPE html>

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

<?php
  $db = new mysqli('localhost', 'root', 'demopw', 'music');
  if ($db->connect_error) {
    die('Failed to connect: '.$db->connect_error);
  }
?>
Creating HTML Output From SQL Query (2)

...<body>

<h1>The following table is retrieved from MySQL:</h1>
<table>
<?php
    $query = 'SELECT * FROM mysongs';
    $result = $db->query($query)
        or die ('Query failed'.$db->error);
    while ($row = $result->fetch_assoc()) {
        echo "\t<tr>\n";
        foreach ($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
    $result->free();
    $db->close();
?
```

```html
</body>
</html>
```
Outlook: Using MongoDB (Document-Oriented)

Heinrichs-MacBook-Pro: hussmann$ mongo
MongoDB shell version: 2.6.5
> use music
switched to db music
> db.mysongs.insert({code:'1', title:'One', artist:'U2',album:'The Complete U2',runtime:272})
WriteResult({ "nInserted" : 1 })

...  
> db.mysongs.find({runtime: {$gt: 250}}, {title: true})
{ "_id" : ObjectId("5448042878b2c1f62e542dc4"),
   "title" : "One" }
{ "_id" : ObjectId("544804cf78b2c1f62e542dc5"),
   "title" : "Wheel in the Sky" }
{ "_id" : ObjectId("5448054978b2c1f62e542dc6"),
   "title" : "Lady in Black" }
{ "_id" : ObjectId("5448054e78b2c1f62e542dc7"),
   "title" : "Smoke on the Water" }
> quit()

JavaScript takes the role of SQL!
Chapter 2: Interactive Web Applications

2.1 Interactivity and Multimedia in the WWW architecture

2.2 Client-Side Multimedia in the Web (Example HTML5)

2.3 Interactive Server-Side Scripting (Example PHP)

2.4 Data Storage in Web Applications (Example Database Access in PHP)

2.5 Integrated Server/Client-Side Scripting (Example jQuery/AJAX)

Literature:


http://jquery.com
Warm-Up (Client-Side Interactive Content)

• Who has already used a library for JavaScript? Which one?
• How does the browser get access to a JavaScript library we are using in our Web pages?
• What is the "DOM"? Why is it important for client-side interactivity?
• We will learn how to use the jQuery library for DOM manipulation. Can we write the same Web apps without using jQuery?
jQuery

- See jquery.com
  - John Resig 2006
- JavaScript Library to assist with
  - traversal and manipulation of HTML through DOM
  - event handling
  - animations
  - Simple AJAX applications (see later)
- Current version: 3.1.1
- jQuery is by far the most used JavaScript library in 2016
  - 02 Nov 2016: 71.4% of all Websites, 96.3% market share in JS libraries (see http://w3techs.com/technologies/overview/javascript_library/all)
- Further libraries build on jQuery (e.g. jQueryUI)
- jQuery is essentially one large JavaScript file
  - included locally or through a delivery network of servers
Using jQuery

• Include the library into any file where jQuery is to be used
  – Locally: `<script type="text/javascript">jquery.js</script>`
  – From jQuery Web site or through various Content Delivery Networks
• jQuery is accessible as a global function and as an object instance
  – Function “jQuery”, abbreviated as “$”
• jQuery includes “Sizzle” engine to traverse and manipulate DOM trees
  – Frequent pattern: `$(selector-expression)`
• jQuery provides additional utility functions
  – Frequent pattern: `$.fname(parameters)`
• jQuery supports event handlers
  – Frequent pattern: `DOMObject.eventname(function)`
  – Convenient pattern: Using local anonymous functions
• jQuery should be executed after DOM tree is ready
  (not necessarily after loading all content)
  – Event handler for `ready` event
Event Handler for jQuery `ready` Event

- Standard places to put jQuery code:
  - in a script block at the end of page
  - executed when DOM tree has been loaded (event handler)

```html
<script src="jquery.js"></script>
<script>
    function runJQuery() {
        alert("run some jQuery code now");
    }

    $( document ).ready(runJQuery);
</script>
```

`jq_init0.html`
Using Anonymous Functions in JavaScript

```javascript
<script>
    function runJQuery() {
        alert("run some jQuery code now");
    }

    $( document ).ready(runJQuery);
</script>

Rewritten with anonymous event handler:

```javascript
<script>
    $( document ).ready(
        function() {
            alert("run some jQuery code now");
        }
    );
</script>
```

Discussion

- What are advantages and disadvantages of using anonymous functions for JavaScript event handlers?
Example: Interactive Highlighting in Table

• Assuming HTML and CSS code for table:

```html
<table>
  <thead>
    <tr>
      <th>#</th>
      <th>Title</th> ...
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>One</td> ...
    </tr>
  </tbody>
</table>
```

```css
<style>
  table    {...}
  th, td   {...}
  thead    {
    background-color: black;
    color: white;
  }
  tr.hilite {
    background-color: grey;
    color: white;
  }
</style>
```
jQuery DOM Selection

• Typical selector arguments for $\$( selector \) 
  – document
  – HTML element names
  – Identifiers (referring to HTML id attribute): #ident
  – CSS classes: .classname
  – Special filters: :filtername

• Path constraints: Space-separated list of selectors
  – Have to appear as (possibly indirect) successors in tree

• Example: Handler hover event on table rows:
  – $\$( 'tr' ).hover(function() { ...hilite... });
  – hover: Handler called on mouseenter and mouseleave events
Understanding jQuery Selection

```javascript
$( 'tr' ).hover(function() {
    ... here some code to switch the CSS class for highlighting ...
});
```

- Which parts of the document will be affected?
- How can we properly un-highlight?
- How can we use jQuery selector syntax to resolve the problems?
- Can CSS syntax be helpful to resolve the problems?
jQuery DOM Manipulation

• jQuery provides functions to
  – modify attributes of HTML elements
  – modify CSS classes attached to HTML elements
  – add or remove parts of the DOM tree
  – retrieve HTML text from DOM tree
  – create DOM tree from HTML strings

• Good practice: Assign/un-assign classes from CSS style sheet
  – Add or remove class:
    \texttt{object.addClass(class)}, \texttt{object.removeClass(class)}
  – Toggle (add/remove) class:
    \texttt{object.toggleClass(class)}

• Example:
  \begin{verbatim}
  \$("#mysongs tbody tr").hover(function() {
    \$(this).toggleClass("hilite");
  });
  \end{verbatim}

jq_table2.html
Example: Extending HTML Table Using jQuery

• Make rows of the table selectable by adding a checkbox column

• jQuery code for table head:

  ```javascript
  $('#mysongs thead tr').
    append('  
    <th>Select</th>
  ');  
  ```

• jQuery code for table body:

  ```javascript
  $('#mysongs tbody tr').
    append('  
    <td style="text-align: center">
      <input type="checkbox" />
    </td>
  ');  
  ```

jq_table3.html
Functional Programming!

• Who remembers functional programming? What are typical languages?
• What is the relationship between functional programming concepts and manipulation of HTML/DOM structures?
Restructuring jQuery Code

• Good practice: Re-use selection results (optimization)
• Apply concepts from functional programming:
  – E.g. `collection.each(fn)`: applies function `fn` to all objects contained in `collection`
• Example:
  ```javascript
  $('#mysongs tbody tr').each(function(){
    $(this).append('  
      <td style="text-align: center">
        <input type="checkbox">
      </td>
    ');
    $(this).hover(function(){
      $(this).toggleClass('hilite');
    });
  });
  ```
Method Chaining

- jQuery: Most functions return an object compatible to the object on which the function was called
- Create *method chains by function composition*
- Simple generic example:
  ```javascript
  $(...).addClass('classname').css(css_prop, css_value);
  ```
- Executing another jQuery query on result set:
  ```javascript
  collection.find('selector');
  ```
- Running example:
  ```javascript
  $(this)
  .append('<td style="text-align: center"><input/ type="checkbox"></td>')
  .find(':checkbox')
  .change(event handler for change event);
  ```
Example: Highlighting Selected Rows in Table

```
.find(':checkbox').change(function(){
    if ($(this).prop('checked')) {
        $(this).parents('tr').addClass('checked');
        numCheckedRows++;
    } else {
        $(this).parents('tr').removeClass('checked');
        numCheckedRows--;
    }
});
```

**parents(element_type):**
Moves upwards in the tree and selects all elements of given `element_type`
Animations in jQuery

• jQuery enables time-dependent transitions
  – between CSS property values
  – adjustable in duration and linearity (“easing” in/out)
• Generic animation method: `animate()`
• Shortcut methods for frequent animations:
  – `show(speed)`, `hide(speed)` for DOM elements
  – simple parameter `speed` with values `slow`, `normal`, `fast`
• Example:
  ```javascript
  if (numCheckedRows==0) $('#btn').show("slow");
  if (numCheckedRows==1) $('#btn').hide("slow");
  ```
• More animations: [https://api.jquery.com/category/effects/](https://api.jquery.com/category/effects/)
Combining PHP, Database Access, jQuery

• jQuery code as part of server page in PHP/MySQL setting
  – jQuery/JavaScript sent from (PHP-enabled) Web server

```html
<body>
  <h1>The following table is retrieved from MySQL:</h1>
  <div style="width: 600px">
    <table id="mysongs" style="width: 600px">
      <thead>...</thead>
      <tbody>
        <!-- PHP code here -->
        <?php
          $query = 'SELECT * FROM mysongs';
          $result = mysql_query($query) ...;
        ... ?>
      </tbody>
    </table>
  </div>
  <input id='btn' type='button' value='...'></input>
</body>
<script src="jquery.js"></script>
<script>
  $(document).ready(function() {...})
</script>
```
Selecting Information Using jQuery/DOM

- Example: Get the IDs of all checked table rows
  - For instance to put them into a shopping cart

```javascript
$('#btn').click(function()
    var selIdsTextArray = $('#mysongs input:checked').map(function()
        return $(this).parents('tr').children().first().text()
    ).toArray();

    ...

});
```

`map` functional
(also from functional programming):
Applying a function pointwise to a collection
Sending Selected Data to Server

- HTTP traditional *synchronous* way:
  - Filling a form, sending a request (GET or POST)
  - Request data: key-value pairs with simple value types
  - Response data: HTML
  - Waiting for response before updating page

- Modern *asynchronous* way ("AJAX"):
  - Sending a request from JavaScript
  - Request and response data:
    String encoding of data structures (e.g. JSON)
  - *Continue script in parallel to waiting for response*

- AJAX is easy with jQuery!
Sending Request Using jQuery

```javascript
$('#btn').click(function(){
    var selIdsTextArray = $('#mysongs input:checked').map(...).toArray();
    var selIdsJson = JSON.stringify(selIdsTextArray);

    $.ajax({
        type: 'POST',
        url: 'serverDummy.php',
        data: {selection: selIdsJson}
    });
});
```
jQuery AJAX Requests with Result

- jQuery `ajax` method
  - (and shorthands `get` and `post`)
  - creates a request to server
- Standard arguments, like:
  - `url`: URL address to send request to
  - `settings`: Key-value pairs (may contain JSON data)
- Example settings:
  - `dataType`: Kind of data expected for answer (e.g. xml, json, html)
  - `success(data, status)`: JavaScript function to be called in case of successful server response
  - `error(requestObj, message, errorObject)`: JavaScript function to be called in case of server response indicating errors
  - `contentType`: Kind of data sent to the server
    - often 'application/json'