Multimedia im Netz
Online Multimedia
Winter semester 2015/16

Tutorial 14 – Major Subject
Today’s Agenda: Exam Preparation

- Announcement
- AJAX
  - Vanilla
  - jQuery
  - Polymer
- NodeJS
  - npm
  - Middleware
- PHP + MySQL
- HTML 5 Canvas
- Theory
Announcements

• Repetition Sheet (assignment 14)
  – for your individual preparation
  – cannot be submitted to UniWorX
  – Try to complete the tasks on paper
  – Do not look up things on the Internet, while you do the tasks. Use your lecture/tutorial material and notes
Exam (1)

• The exam is open book
• Bring:
  – handwritten notes, printed slides, code, books (if necessary)
  – valid student card
  – valid personal ID card

• Leave at home:
  – any electronic device. That includes calculators of any kind, phones, computers, smart watches, etc.
Exam (2)

• You need to register on UniWorX to participate:
  – choose “Major Subject (Hauptfach)”
  – Registration hard deadline: 08.02.2016 10:00
  – De-registration hard deadline: 09.02.2016 10:00
• If you do not de-register and not show up, the attempt will be counted as “failed”.

• Date: **11.02.2016 10:00 – 12:00** (90 minutes writing time)
• Location: M118 & A240Hauptgebäude, Geschwister-Scholl-Platz 1

• Retry exam probably early April, but we’d prefer if you passed the first exam 😊
• Grades will take a while. Don’t write hasty emails.
Breakout: Questions!

• Take *5 minutes* time to write down questions
• Topics:
  – anything from the tutorial
  – anything from the assignments
• We are going to collect your questions and answer as many as possible during this tutorial.
AJAX
AJAX Basics

- Acronym: **Asynchronous JavaScript And XML**
- Allows passing around data between client- and server-side applications back and forth – **without refreshing the page**
- AJAX requests (also: XHR = XMLHttpRequest):
  - **GET**: retrieve data – no manipulation on the server
  - **POST**: modify data (and/or retrieve data)
- Nowadays XML mostly replaced by JSON (in JavaScript environments)
Example Data: artists.json

- Let’s fetch this asynchronously

```json
[
  {
    "name": "Arcade Fire",
    "members": [
      "Win Butler",
      "Régine Chassagne",
      "William Butler",
      "Richard Reed Parry",
      "Tim Kingsbury",
      "Jeremy Gara"
    ]
  },...
]
```

- The data should then be displayed as heading (band name) and unordered list (members)
var req = new XMLHttpRequest();
req.onreadystatechange = function(){
    var artists;
    var vanillaOutput = document.querySelector('#vanilla-output');

    if(req.readyState == 4 && req.status == 200){
        artists = JSON.parse(req.responseText);
        artists.forEach(function(req(artist){
            var heading = document.createElement('h3');
            var list = document.createElement('ul');
            heading.innerHTML = artist.name;
            artist.members.forEach(function(member){
                var item = document.createElement('li');
                item.innerHTML = member;
                list.appendChild(item);
            });
            vanillaOutput.appendChild(heading);
            vanillaOutput.appendChildChild(list);
        });
    }
}
req.open('GET','artists.json',true);
req.send();
$.get('artists.json', function(artists){
    var jqueryOutput = $('#jquery-output');
    $(artists).each(function()
    {
        var heading = $('h3');
        var list = $('ul');
        heading.html(this.name);
        $(this.members).each(function()
        {
            var item = $('li');
            item.html(this);
            list.append(item);
        });
        jqueryOutput.append(heading);
        jqueryOutput.append(list);
    });
});
AJAX with Polymer

```html
<template is="dom-bind">
  <iron-ajax url="artists.json"
    auto
    last-response="{{artists}}">
  </iron-ajax>
  <template is="dom-repeat"
    items="[[artists]]">
    <h3>[[item.name]]</h3>
    <template is="dom-repeat"
      items="[[item.members]]">
      <li>[[item]]</li>
    </template>
  </template>
</template>
```
NodeJS + Express
NodeJS Basics

• What is it?
  “Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.” (official website, nodejs.org)

• Node apps are JavaScript files! → Server-side JavaScript

• Why do we want it?
  – non-blocking I/O
  – scalable
  – web-apps act as standalone web-server
  – largest ecosystem of open source libraries
npm

• Node package manager.
• package.json:
  – Package-information
  – dependencies
• Common usage:
  npm install --save body-parser
• Install packages globally:
  npm install -g bower
• Run a script:
  npm run start

```
{
  "name": "mmn-examples",
  "version": "0.0.0",
  "description": "examples",
  "scripts": {
    "start": "node bin/www"
  },
  "author": "Tobias Seitz",
  "license": "MIT",
  "dependencies": {
    "body-parser": "^1.14.2"
  }
}
```
Express

- Web app framework for NodeJS
- Can be set up with the **express-generator** package
- Features:
  - Middleware
  - Routing
  - Templates
- Currently the most popular framework for NodeJS
Middleware

- A middleware is a function that sits between the request and the response (“in the middle”)
- Does something with the request. Typical tasks:
  - parse the HTTP message body to a JSON object (*body-parser*)
  - parse cookies (*cookie-parser*)
  - authenticate the user and (dis)allow a request
- Usually more than one middleware per route (middleware chain)

```javascript
app.use('/', function(req, res, next){
  // do something with req object
  // then either send the response or call:
  next();
});
```
app.use(function(req, res, next) {
    if (req.query.secret == 'mmn' || req.body.secret == 'mmn') {
        req.isAuthenticated = true;
    }
    next();
});
app.use('/myartists', function(req, res) {
    if (req.isAuthenticated) {
        res.sendFile(path.join(__dirname, '../artists.json'));
    } else {
        res.json({'message': 'sorry, you are not allowed to go here.'})
    }
});
NodeJS: Your Questions (Assignment 07)

• What functionality goes into our custom module spotify.js, what goes into index.js?
  – The idea is that index.js serves as a routing module. Its “concerns” are therefore: include other modules, register routes for them

• Which of these contain the middleware?
  – both, because the `router object is middleware`
  – index.js: `router.use('/spotify', spotify);`
  – spotify.js:
    ```javascript
    router.use('/', express.static(path.join(__dirname, '../spotifysearch')));
    ```

• Middleware function: `function (req, res[, next]) {...}
  – You see middleware functions in:
    • app.use(...), app.get(...), app.post(...), ...
    • router.use(...), router.get(...), router.post(...)```
PHP & MySQL
PHP Example: “Add my number”
What’s the problem?

```html
<!DOCTYPE html>
<html><head lang="en"><meta charset="UTF-8">
  <title></title></head><body>
<?php
if(isset($_POST['add'])) {
    $currentCounter = 0;
    echo "$currentCounter + " . $_POST['add'] . " = " . ($currentCounter + $_POST['add']);

    $currentCounter += $_POST['add'];
}
?>
<form method="POST" action="add-my-number.php">
  <input type="number" placeholder="" name="add">
  <input type="submit" name="submit" value="Add">
</form>
</body></html>
```
Breakout: Code-Along PHP

• Fix the code from the previous slide.

• When the form is submitted, the number from the input field should be added to the previous result.

• Timeframe: 10 minutes
PHP + MySQL

• Multiple functions and APIs available for PHP to work with databases:
  – `mysql` („Deprecated“ since PHP 5.5.0)
  – `mysqli` (i is for „improved“)
  – PDO (PHP Data Objects)
• „mysql“ is still supported for older PHP versions
• It is highly recommendable to use mysqli or PDO
mysqli (object oriented)

- Establish connection
  
  ```php
  $c = new mysqli("host","user","password","db");
  ```

- PHP statement for MySQL query
  
  ```php
  $results = $c->query($query);
  ```

- Process the results
  
  ```php
  $results->fetch_assoc();
  $results->fetch_row();
  ```

  ```php
  $results->fetch_all(MYSQLI_BOTH);
  $results->fetch_all(MYSQLI_ASSOC);
  $results->fetch_all(MYSQLI_NUM);
  ```

- Close the connection
  
  ```php
  $c->close();
  ```
### mysqli Example

```
<?php
$host = 'localhost'; $user = 'mmn1516';
$password = 'mmnpassword'; $database = 'mmn1516';
$c = new mysqli($host,$user,$password,$database);

$now = date('Y-m-d H:i:s');
$queryString = "INSERT INTO expenses
    (amount,reason,person,spending_date)
    VALUES (25,'coffee beans','Max',"$now")";
$c->query($queryString);

$queryString = "SELECT * FROM expenses"
$results = $c->query($queryString);

while($row = $results->fetch_assoc()){
    echo
    $row['id']. '.' . $row['amount']. '.' .
    $row['reason']. '.' . $row['person']. '.' .
    $row['spending_date']. '<br />';
}
?>
```
When do we use “regular” mysqli queries, when do we need prepared statements?

- Both are fine in pretty much all situations.
- Differences:
  - Prepared statements require a little more effort
  - Regular mysqli queries tend to be less secure (SQL injections)
HTML5
Quiz: HTML5

• Name 3 elements that are ‘new’ in HTML5!
• Which document type is correct for HTML5:
  a)  <!DOCTYPE html>
  b)  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 5.0//EN" "http://www.w3.org/TR/html5/strict.dtd">
  c)  <!DOCTYPE HTML5>
• onblur and onfocus are...
  a)  elements
  b)  event attributes
  c)  style attributes
• Which attribute of <script> is no longer required?
  rel | href | src | type

http://www.w3schools.com/quiztest/quiztest.asp
var x,y;
var canvas = document.getElementById('canv');
var context = canvas.getContext('2d');
var columns = 3, rows = 4, size = 20, offset = 5;

for(var i = 0; i < columns; i++){
  for(var j = 0; j < rows; j++){
    x = i * (size + offset), y = j * (size + offset);

    context.beginPath();
    context.fillStyle = i % 2 == 0 ? '#0000ff' : '#ff0000';

    context.strokeStyle = '#000000';
    context.lineWidth = "2";

    context.rect(x,y,size,size);
    context.stroke();
    context.fill();
  }
}

Ludwig-Maximilians-Universität München

Online Multimedia WS 2015/16 - Tutorial 14 - 29
Assignment 13 - HbbTV 2.0

• Most anticipated features:
  – HTML5 user experiences
  – UltraHD
  – Seamless viewing across TV, tablet, smartphone
  – Second screens (interactivity, backchannels)

• HEVC = High efficiency video coding
  – Known as H.265 / MPEG-H 2
  – Successor to H.264 / MPEG-4-AVC
  – Competitor to VP9 (respectively VP10)

https://en.wikipedia.org/wiki/High_Efficiency_Video_Coding
MPEG-DASH

- DASH: Dynamic Adaptive Streaming over HTTP
- Video files are available as fragments in different resolutions.
- 1st step: Client requests manifest file with available options
- Key feature: Adaptive Bitrate (ABR)
  - reduces buffering by requesting fragment with the best quality that can be transferred without video-stalling (freeze)
  - reduces traffic because lower quality is requested automatically
- Article:
  - Heavily congested networks: many clients on the same network
  - YouTube’s HTML5 player is DASH compatible → automatic selection of best possible quality
The GitHub Contributors - Thanks!!
Tutors wanted!

• If you enjoyed this course and are excited about the topic, become a tutor in the next winter semester!

• Responsibilities:
  – run tutorials (at least one)
  – help with corrections
  – help students with programming

• Feel free to contact us for further details 😊 tobias.seitz@ifi.lmu.de
Thanks for joining!
Good luck for the exam!