Multimedia im Netz

Online Multimedia

Winter semester 2015/16

Tutorial 04 – Major Subject

Today's Agenda

- AJAX with plain Javascript
- jQuery
  - DOM Operations
  - Event Handling
  - AJAX
- Quiz

Semester Plan: current State

Dates	Topics
19.10. & 21.10.	Organization, Client-Side Scripting, git
26.10. & 28.10.	Server-side scripting with PHP – Basics
02.11. & 04.11.	PHP: Sessions and Data Storage
09.11. & 11.11.	PHP & MySQL, AJAX
16.11. & 18.11.	**AJAX, jQuery**
23.11. & 25.11.	*More AJAX, more jQuery, Advanced JavaScript*
30.11. & 02.12.	**Open Lab Day** + NodeJS Basics, Express Framework
07.12. & 09.12.	NodeJS: Routing, Database Access
14.12. & 16.12.	NodeJS: Authentication
21.12. & 23.12.	Christmas Tutorial – Programming Consultation
11.01. & 13.01.	Digital Rights – Watermarking Techniques
18.01. & 20.01.	Multimedia Content Description, Introduction to AngularJS
25.01. & 27.01.	AngularJS2, Webcomponents with Polymer
01.02. & 03.02.	Repetition / Cancelled (depending on final exam date)

AJAX

AJAX

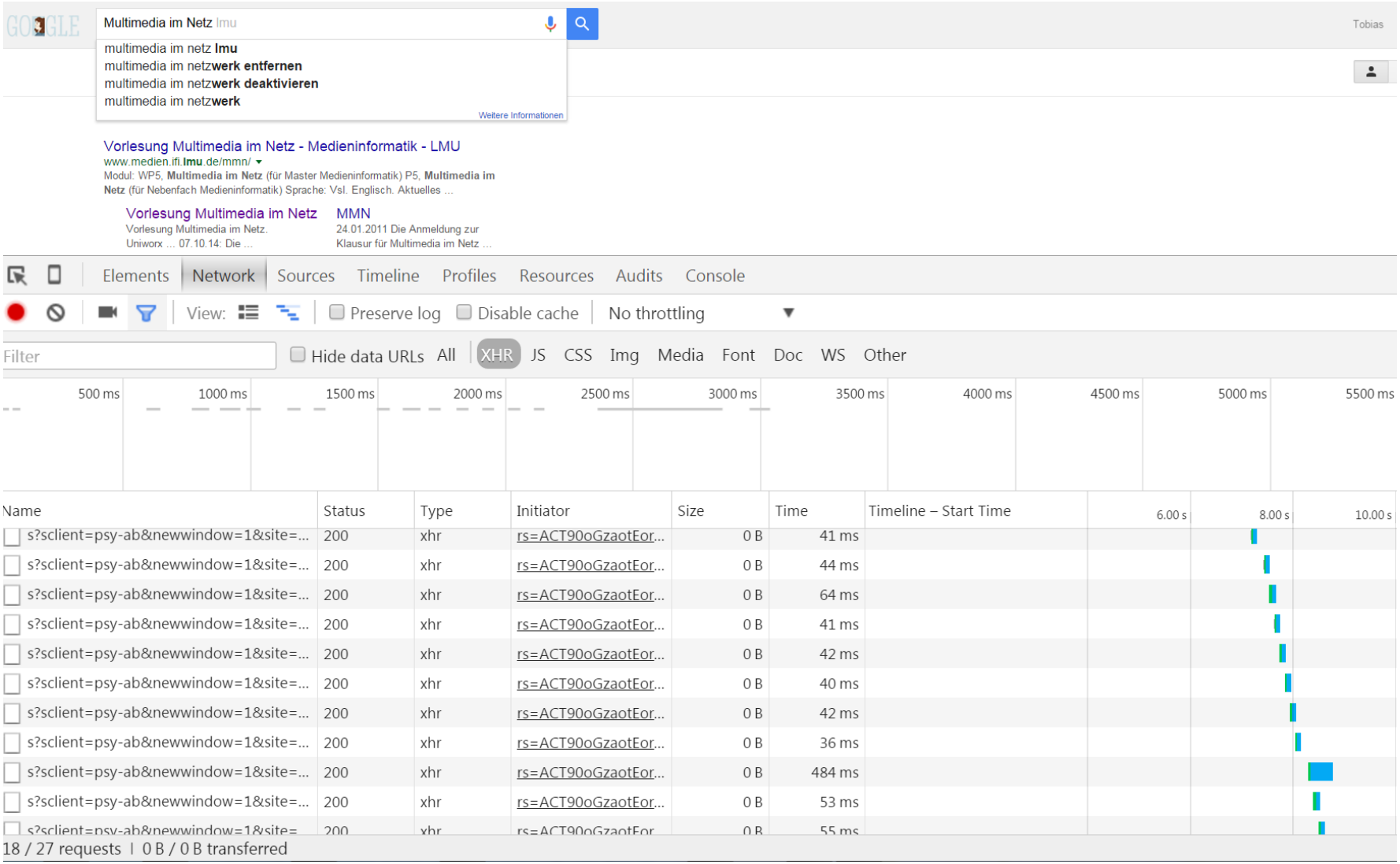
- Acronym: **A**synchronous **J**avaScript **A**nd **X**ML
- Allows passing around data between client- and server-side applications back and forth – **without refreshing the page**
- AJAX requests (also: XHR = XMLHttpRequest):
  - GET: retrieve data – no manipulation on the server
  - POST: retrieve and/or modify data

XMLHttpRequest

- API to transfer data between client and server without a full page refresh
- Originally designed by Microsoft, adopted by all other browsers
- Ready States 0, 1, 2, 3, 4
  - Mostly relevant: **4** (done)
  - See lecture slides
- HTTP Status Codes are accessible in XMLHttpRequest.status
  - 200: Success
  - 401: unauthorized
  - 404: not found
  - 500: internal server error

https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

AJAX Requests in the Dev Console



Example: Fetching HTML asynchronously

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>AJAX request with plain AJAX</title>
</head>
<body>
<div id="response"></div>
<script>
    var request = new XMLHttpRequest();
    request.open('GET','../php/htmlResponse.php');
    request.onreadystatechange = function(){
        var outputDiv = document.getElementById('response');
        if(request.readyState != 4) return;
        outputDiv.innerHTML = request.responseText;
    };
    request.send();
</script>
</body>
</html>
```

ajax/ajaxPlainJS.html

~~XML~~ JSON!

- Before JSON was widely adopted, XML was the state of the art
- **J**ava**S**cript **O**bject **N**otation
- Human-readable format for data exchange
- Based on key-value pairs

- **example:**

```
{
  "firstName": "John",
  "lastName": "Doe",
  "phone": [
    {
      "type": "Home",
      "number": "5648978965"
    },
    {
      "type": "Mobil",
      "nummer": "6458979878"
    }
  ]
}
```

Example: JSON from PHP Script

```php
<?php

header('Content-Type: application/json');

$artists =
    array("artists"
        =>array("The Rolling Stones",
            "The Beatles",
            "Red Hot Chili Peppers",
            "Käptn Peng"));

$jsonArtist = json_encode($artists);

echo $jsonArtist;
?>
```

php/jsonResponse.php

Example: Handling a JSON Response

```
<!DOCTYPE html>
<html lang="en">
<head><meta charset="UTF-8">    <title>Loading JSON data</title>
</head><body>
<div id="response" style="white-space: pre"></div>
<script>
    function prettyJSONString(jsonObj){
        return JSON.stringify(jsonObj,null,4);
    }
    var request = new XMLHttpRequest();
    request.open('GET','../php/jsonResponse.php');
    request.onreadystatechange = function(){
        var jsonResponse;
        var outputDiv = document.getElementById('response');
        if(request.readyState != 4) return;
        jsonResponse = JSON.parse(request.responseText);
        outputDiv.innerText = prettyJSONString(jsonResponse);
    };
    request.send();
</script>
</body></html>
```

ajax/ajaxLoadJSON.html

jQuery

jQuery: Basic Information

- http://jquery.com
- JavaScript-Library, currently at version 1.11.3 or 2.1.4
- Features:
  - Easy DOM-access
  - Convenient event-handling
  - Animations
  - Simplified AJAX-Requests
  - Cross-browser compatibility

- „jQuery is designed to change the way that you write JavaScript“
- But don’t get confused: jQuery **IS STILL** JavaScript!

Including the jQuery library

- Only a single JavaScript file is required
- 3 Options:
  - Download and include locally (offline):

```
<script src="jquery-1.11.4.min.js"
        type="text/javascript"></script>
```

  - Hotlink (Google APIs)

```
<script
src="//ajax.googleapis.com/ajax/libs/jquery/2.1
.3/jquery.min.js"></script>
```

  - Hotlink (jQuery Site):

```
<script src="//code.jquery.com/jquery-
2.1.3.min.js"></script>
```

Basics

- Global jQuery function: `$()`
- Parameters: any CSS Selector
- Returns: jQuery-object, that offers handy methods.

- Examples:
  - `$("#myDiv")`
  - `$("div.container")`
  - `$("input[type='text']")`

- `$` is also an object that offers additional methods, e.g. `$.inArray(value,array)`

DOM-Manipulation (1)

```
<div id="myDiv">Container</div>
```

- Get the HTML content of an element:
  ```
  var content = $('#myDiv').html();
  ```

- Modify the HTML content of elements:
  ```
  $('#myDiv').html("<span>My New Content</span>");
  ```

- **Note:** `.html()` can be used as both getter and setter!

DOM-Manipulation (2)

```
<div id="myDiv">Container</div>
```

- Create nodes and add them to the DOM:
```
$("#myDiv").after("<span>More Content</span>");
```

- Get/set attributes:
```
$("#myDiv").addClass("container");
$("#myDiv").attr("id", "newID");
```

- More examples:
http://api.jquery.com/category/Manipulation/

Breakout – part 1

- Download the file “breakout.html” from the GitHub repo
- Use jQuery to solve this task.
- Tasks:
  1. Dynamically add the class “rick” to all buttons in the right column.
  2. Add four buttons containing the text:
     1. run
     2. around
     3. and
     4. desert
  3. The new buttons should also have the “rick” class

- Time frame: 20 Minutes

Event-Handling (1)

- Event-Handler: get notified about certain events (e.g. clicks) and proceed to execute a given action (= callback)

- jQuery click handling:

```
$("#myID").click(function(e){ ... });
```

  Alternative:

```
$("#myID").on("click", function(e){ ... });
```

- **Note:** Event Handlers only work on elements that have been added to the DOM, i.e. don‘t add the handler before adding the element to the DOM.

Event-Handling (2)

- There are a lot of event-handlers:
  click, change, focus, submit, keypress, …
  http://api.jquery.com/category/Events/

- Prevent the default handler: `preventDefault();`

```
$("a").click(function(event){
    event.preventDefault();
    alert("Link: " + $(this).attr("href"));
});
```

Breakout – part 2

- Extend the script from earlier
- Goal: The user clicks buttons and creates the lyrics to the song “Never gonna give you up” by Rick Astley.
  - Add event listeners to the buttons.
  - If a button was clicked, its text is copied to the #lyrics container

- Time frame: 15 Minutes

AJAX and jQuery

- jQuery offers methods for AJAX-requests:
  - **`$.ajax()`**
  - convenience methods: **`$.get()`**, **`$.post()`**, **`$.load()`**…

- Advantages:
  - ease-of-use
  - readability
  - cross-browser compatibility

Example: Asynchronous Contact Form

```
<!DOCTYPE html>
<html lang="en">
<head><meta charset="UTF-8"><title>Asynchronous Contact Form</title>
    <style>label{ width: 100px;display: block;}</style>
</head><body>

<form id="contactForm">
    <label>First Name:
    <input type="text" name="firstName" /></label>
    <label>Last Name:<input type="text" name="lastName" /></label>
    <label>Message:
        <textarea name="message" cols="50"></textarea></label><br />
    <input type="submit" value="Submit!"/>
</form>

<div id="response" style="display: none;"></div>

<script src="https://code.jquery.com/jquery-2.1.3.min.js"></script>
<script>
    // ... content from next slide.
</script>
</body></html>
```

Example: Submitting the Form with Data

```
$(document).ready(function(){

   $('#contactForm').submit(function(event){
      var self = $(this);
      var data = self.serialize();

      event.preventDefault();
      // ajax request on next slide!
   });
});
```

Example: Request & Response

```
$.ajax({
    url: '../php/contactFormHandler.php',
    type: 'POST',
    data: data,
    success: function(response){
        var output = $("#response");
        response = (response instanceof String) ?
                $.parseJSON(response) : response;

        self.fadeOut(function(){
            output.html('<p>Thanks for your message, '
                    +response.info.firstName+'!</p>');
            output.fadeIn();
        });
    }
});
```

PHP Script: contactFormHandler.php

```php
<?php
if (isset($_POST['firstName']) &&
    isset($_POST['lastName'])) {

    $response =
            array("status" => "OK",
                "info" => array(
                    "firstName" => $_POST['firstName'],
                    "lastName" => $_POST['lastName']));
    if(isset($_POST['message'])) {
        // this is where you do something
        // with the message, e.g. send it by email.
    }
} else {
    $response = array("status" => "missingParameter");
}

header("Content-type: application/json");
echo json_encode($response);
?>
```

Breakout – part 3

- Extend the script from earlier
- Stick to jQuery for this task
- Goal: The lyrics are validated by a server script.
  - Retrieve the composed text from the #lyrics div
  - Send it to a server script
  - The script validates the order of the words and response with a JSON object
  - Inform the user if the words were correct by re-coloring the background of the #lyrics div.

- Time frame: 15 Minutes

Some more details.

AJAX Request Debugging

- To send requests manually (not via a script), you can use a number of tools
- For Chrome:
  - Postman (also as standalone app)
  - REST Console
- For Firefox:
  - REST Client
  - RESTeasy

Same Origin Policy (SOP) – Part 1

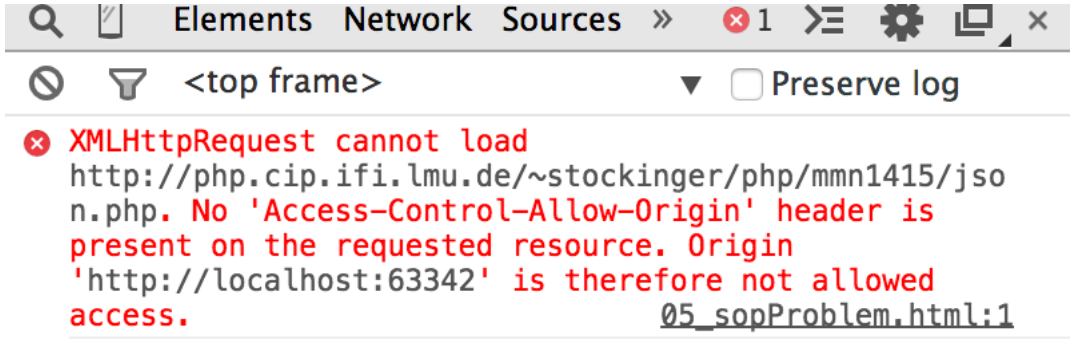
- Example: your script runs is executed at http://example.com

- Same origin:
  - http://example.com
  - http://example.com**/**
  - http://example.com**/another/page**

- Different origin:
  - http://**www.**example.com
  - http://example.com**:3830**/
  - http**s**://example.com
  - http://example.**org**

Same Origin Policy (SOP) – Part 2

- SOP tries to hamper **cross site scripting**
- **Error message example:**



- SOP would block AJAX request to foreign domains entirely ☹
- It will become apparent once you start using your backend scripts remotely
- Solutions:
  - Cross-Origin Resource Sharing (CORS)
  - JSON with Padding (JSONP)
  - Apache Reverse Proxies

Round-up Quiz

1. What does the Acronym AJAX stand for?
2. Explain what “asynchronous” means!
3. What is the difference between GET and POST?
4. Which “readystate” is the most important one for XMLHttpRequests?
5. What is an advantage of JSON compared to XML?
6. What is json_encode() good for?
7. Name an example usage for event.preventDefault()?
8. What does form.serialize() do?

Thanks!

What are your questions?

Let’s begin with the Assignment!

- Download the assignment sheet
- Start with task 1
- You can collaborate with your neighbor
- The task builds on a previous assignment, so we provide a sample solution.

- Turn in the assignment by November 23rd, 12:00 noon via UniWorX