# Multimedia im Netz (Online Multimedia)

## Wintersemester 2014/15

Übung 06 (Hauptfach)

# Today‘s Agenda

- Flashback: 5th Tutorial
- Announcements
- jQuery: AJAX
- Code-A-Long: Filtering a Movie List

## Flashback!

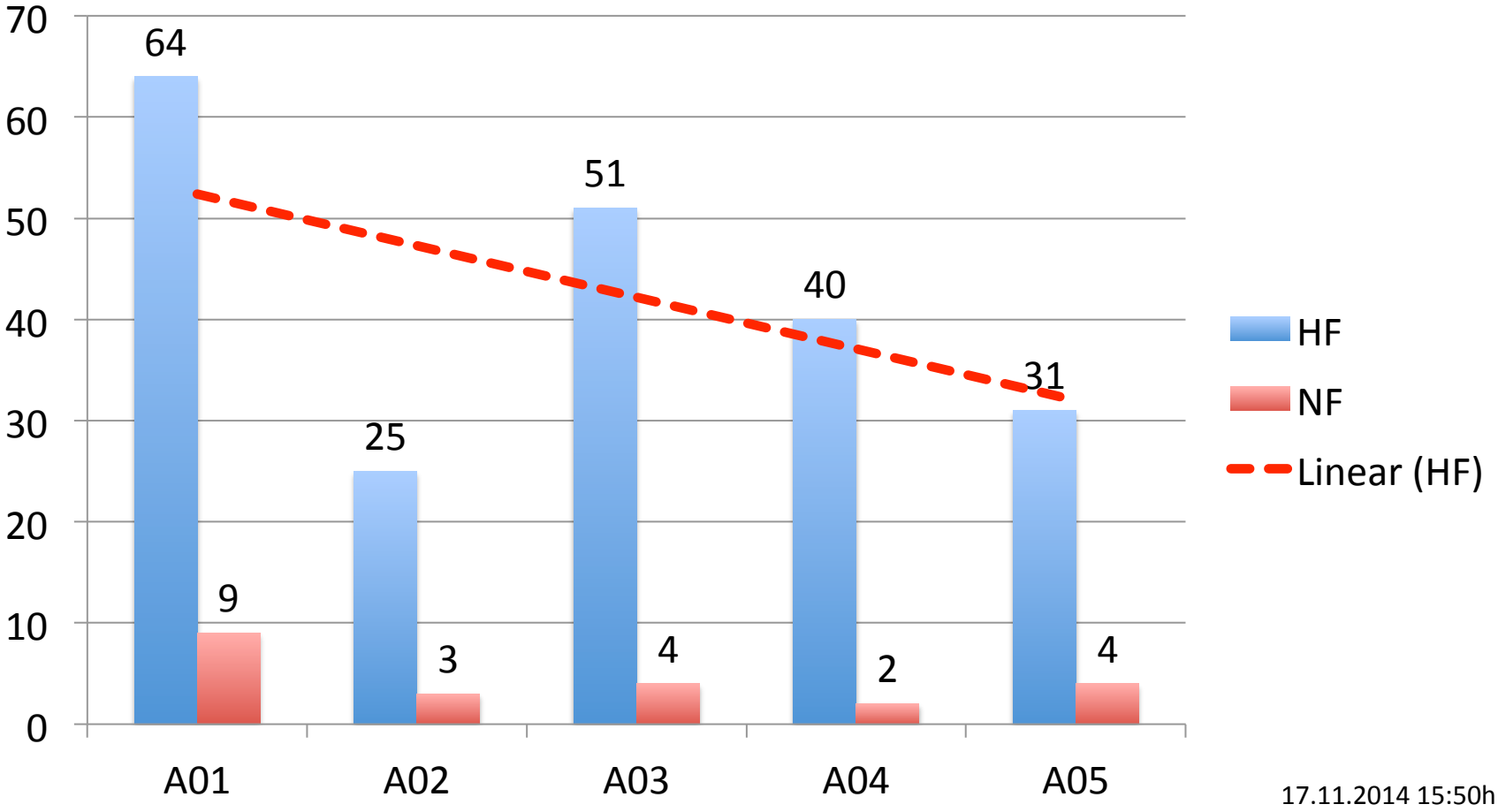
Explain a concept from last week‘s tutorial to your grandmother!

# Announcements

- Assignment 06
  - Assignment 06 is due in 2 weeks.
  - Tutors will help you with your programming assignment next week and answer your questions.
  - Please review the learning material and your assignments. Prepare questions, and send them to tobias.stockinger@ifi.lmu.de in advance.

- Evaluation
  - There is going to be a small evaluation of the tutorials during the next two weeks.
  - You benefit from your constructive feedback right away

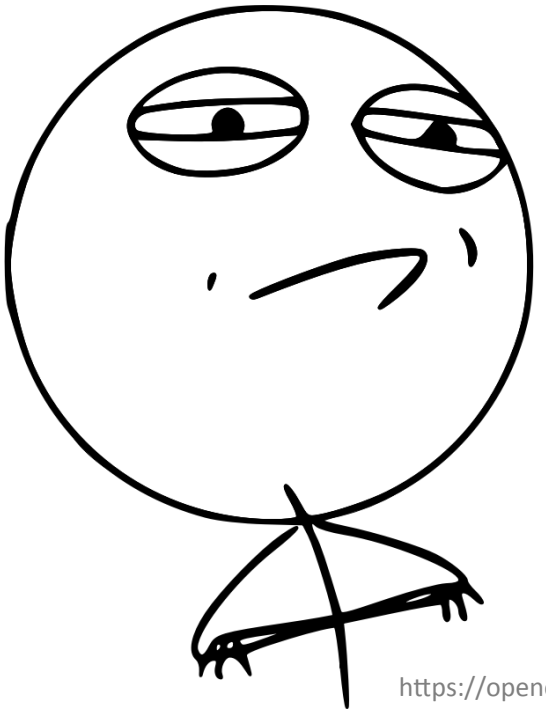
# Assignment Submissions



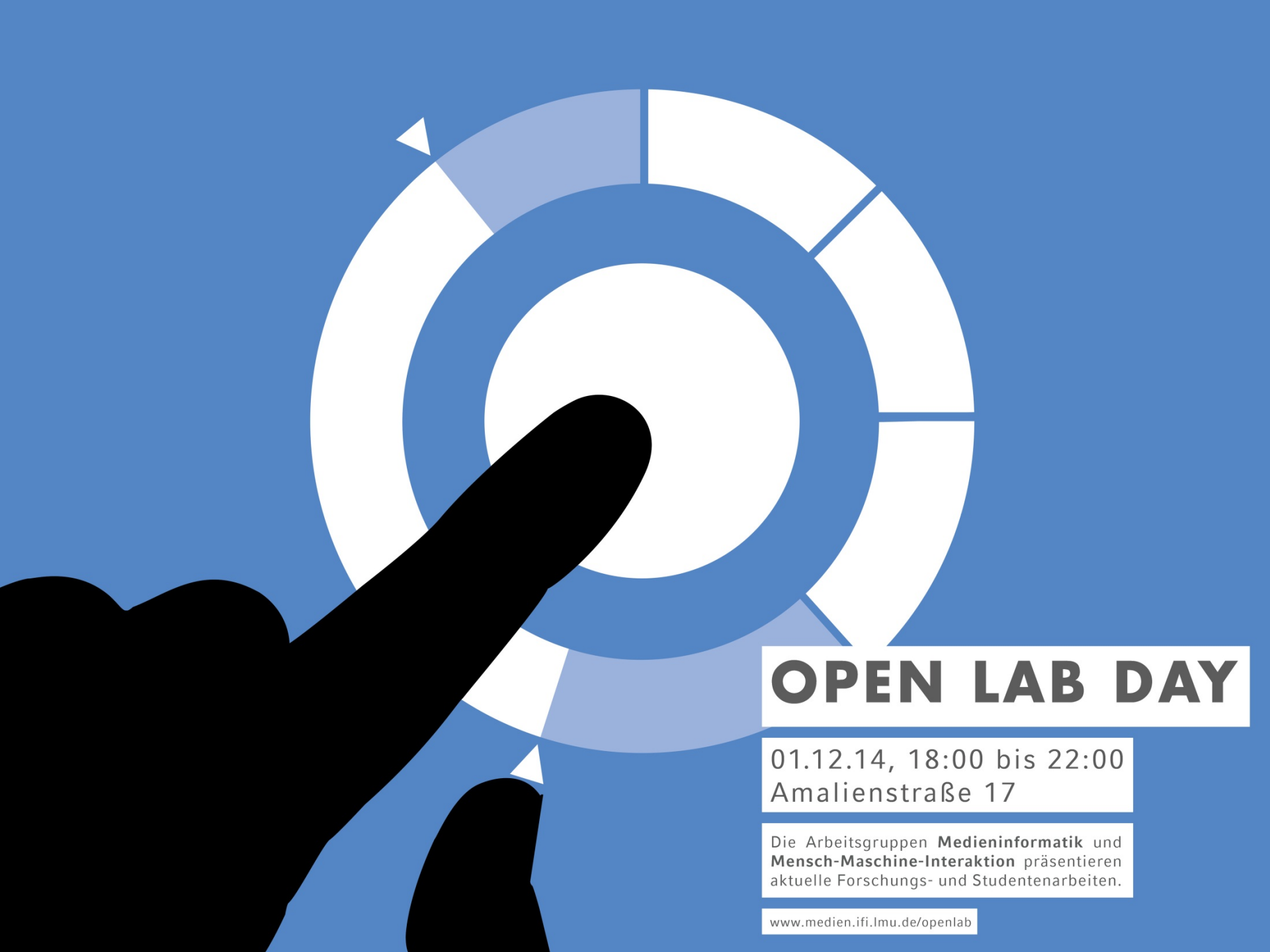

17.11.2014 15:50h

# Challenge

- If more than **75** students (HF & NF) submit assignment 06, there will be a short **mock exam** prior to the final exam.

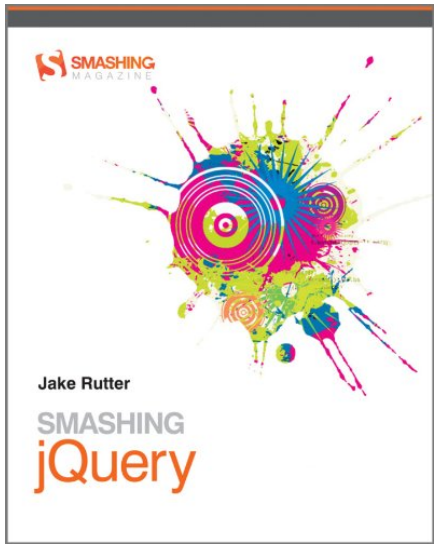


https://openclipart.org/detail/168636/challenge-accepted-by-tavin


OPEN LAB DAY
01.12.14, 18:00 bis 22:00
Amalienstraße 17
Die Arbeitsgruppen **Medieninformatik** und **Mensch-Maschine-Interaktion** präsentieren aktuelle Forschungs- und Studentenarbeiten.
www.medien.ifi.lmu.de/openlab

# jQuery Books

- Rutter, Jake. Smashing JQuery. John Wiley & Sons, 2011.
- Duckett, Jon. JavaScript and JQuery: Interactive Front-End Web Development. John Wiley & Sons, (2014).




# AJAX (ˈeɪdʒæks)

- **A**synchrounous **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 = XML HttpRequest):
  - GET: retrieve data – no manipulation on the server
  - POST: retrieve and/or modify data

# AJAX and jQuery

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

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

# Loading HTML Content

- HTML Content from the same site or server can be loaded into certain elements with `$.load()`
- Example

```
function init(){
    // this only works with local resources!
    $("#container").load('bavariaIpsum.html');
}
$(document).ready(init);
```

# Example: Loading an XML File

```
$.ajax({
    url : 'movies.xml',
    success : function(data){
        // data = xml document
        $("body").append(data.childNodes);
    }
});
```

$.ajax() documentation: https://api.jquery.com/jQuery.ajax/

# Example: Loading an XML File

```
$.ajax({
    url : 'movies.xml',
    success : function(data){
        // data = xml document
        $("body").append(data.childNodes);
    }
});
```

Callback function.
Once the data has been loaded, this function is called.
It can take a while for this to be called – why?

$.ajax() documentation: https://api.jquery.com/jQuery.ajax/

# Parameters: Example (I)

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="UTF-8">
    <title>API Test</title>
    <script src="jquery.js"></script>
    <script>
        // our code
    </script>
</head>
<body>
<form id="contactForm">
    <label for="firstname">First Name:</label>
    <input type="text" id="firstName" name="firstName" />

    <label for="lastname">Last Name:</label>
    <input type="text" id="lastName" name="lastName" />

    <input type="submit" value="Submit!"/>
</form>
<div id="result">
    <!-- all asynchronously loaded content will be put here -->
</div>
</body>
</html>
```

# Parameters: Example (II)

```
<script>
    function init(){

        function successCallback(data){
            $("#result").html(data);
        }
        $("#contactForm").submit(function(event){
            event.preventDefault(); // if we don't do this, the page is actually
            var self = $(this); // this --> #contactForm
            var firstName = self.find("#firstName").val();
            var lastName = self.find("#lastName").val();
            $.ajax({
                url : 'apiScript.php', // the script we're calling
                type: 'POST', // the method we use
                data : { // the data we pass to the script
                    firstName : firstName,
                    lastName : lastName
                },
                dataType : "text", // the expected response data type
                success : successCallback // what to do if we succeed
            });
        });
    }
    $(document).ready(init);
</script>
```

# PHP Script: apiScript.php

```php
<?php

$firstName = $_POST['firstName'];
$lastName = $_POST['lastName'];

echo "First Name: $firstName <br />";
echo "Last Name: $lastName";
```

# .serialize()

```
<script>
    function init(){

        function successCallback(data){
            $("#result").html(data);
        }

        $("#contactForm").submit(function(event){
            event.preventDefault(); // if we don't do this, the page is actually
            var data = $(this).serialize();
            $.ajax({
                url : 'apiScript.php', // the script we're calling
                type: 'POST', // the method we use
                data : data, // the serialized data
                dataType : "text", // the expected response data type
                success : successCallback // what to do if we succeed
            });
        });
    }
    $(document).ready(init);
</script>
```

# DataType

- The DataType field allows us to specify the type of data that we **expect from the server**:
  - Text
  - HTML
  - Script
  - JSON
  - …

# JSON

- **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: AJAX-Request

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

    // anonymous submit handler
    $("#contactForm").submit(function(event){
        // this prevents the page from refreshing
        event.preventDefault();

        // make an object from our form data.
        // use console.log(data) to inspect it.
        var data = $(this).serialize();

        $.ajax({
            url: "json.php",
            type: "POST",
            data: data,
            dataType: "text",
            // callback with response data
            success: function(response){
                // we fill this out later.
            }
        });
    });
});
```

# Example: PHP-Script json.php

```php
<?php

// test script to send a response in json format.
$data = $_POST;

// we just return all parameters as json
echo json_encode($data);
```

# Example: $.parseJSON(data)

```
// callback with response data
success: function(response){
    var output = $("#output");
    var jsonResponse = $.parseJSON(response);

    // iterate over key/value pairs and
    // append each pair to the output container
    $.each(jsonResponse,function(key,value){
        // create a container
        // use @key for its id
        // add some text
        var child = $('<div />').html(key+': '+value);
        child.appendTo(output);
    });
}
```

# AJAX 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

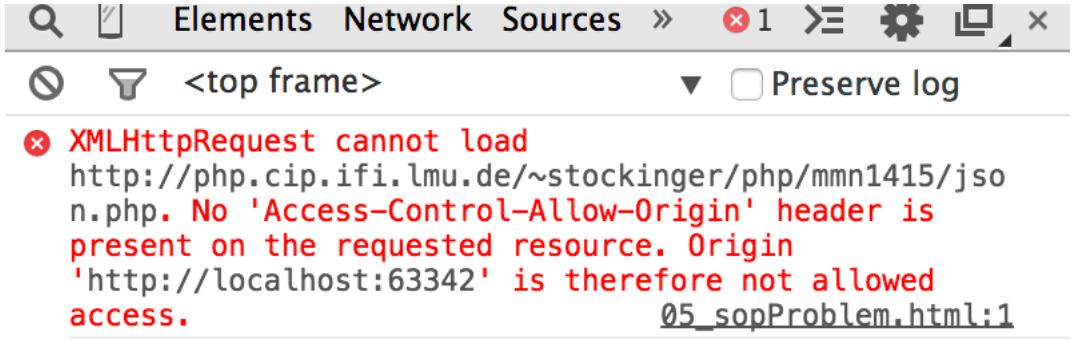
- Example: your script runs is executed at http://example.com

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

- Different origin:
  - http://www.example.com
  - http://example.com:3830/
  - https://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 Mashups entirely ☹
- It will become apparent once you start using your backend scripts remotely
- Solutions for Mashups:
  - Cross-Origin Resource Sharing (CORS)
  - JSON with Padding (JSONP)
  - Apache Proxies

## Fast forward

Write down

**1 thing**

that you have learned today.

# Assignment 6

- **Topic: Currency Converter**
- **Due in:** 2 Weeks
- **Due date:** 01.12.2014
- **Programming consulting:** 24.11.2014 (no or very few slides, no input, just programming help)

# Thanks!

# What are your questions?