Multimedia im Netz (Online Multimedia)

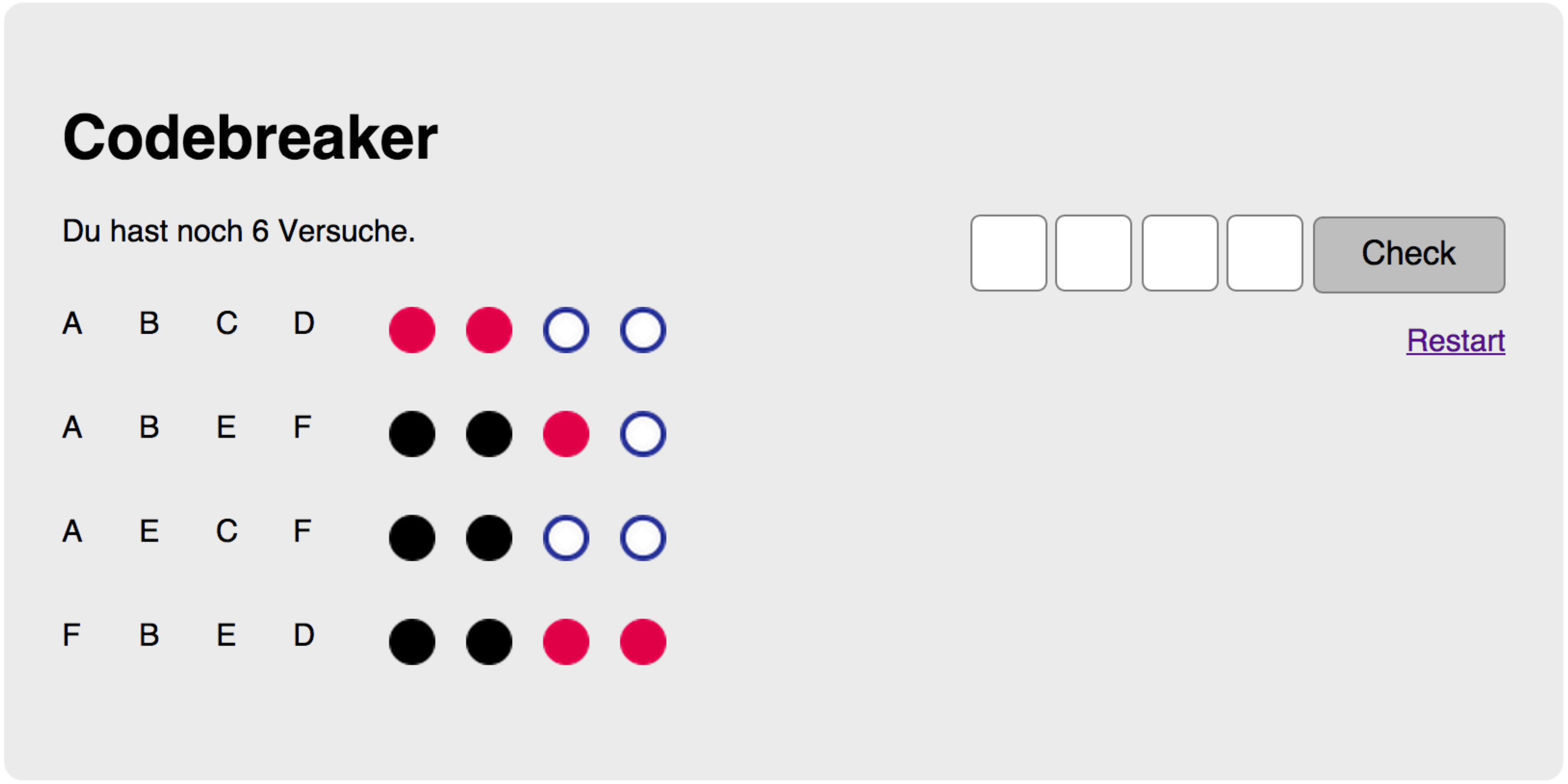
Wintersemester 2014/15

Übung 03 (Hauptfach)

Today‘s Agenda

- PHP Assignments:
  - Discuss „Codebreaker“ Solution
  - Discuss „Gallery“ Solution
- Introduction to HTML 5
  - New Elements
  - Canvas
- Javascript
  - OOP in JavaScript
  - Closures
  - Debugging

Codebreaker



Du hast noch 6 Versuche.

A	B	C	D	🔴	🔴	○	○
A	B	E	F	●	●	🔴	○
A	E	C	F	●	●	○	○
F	B	E	D	●	●	🔴	🔴

Gallery

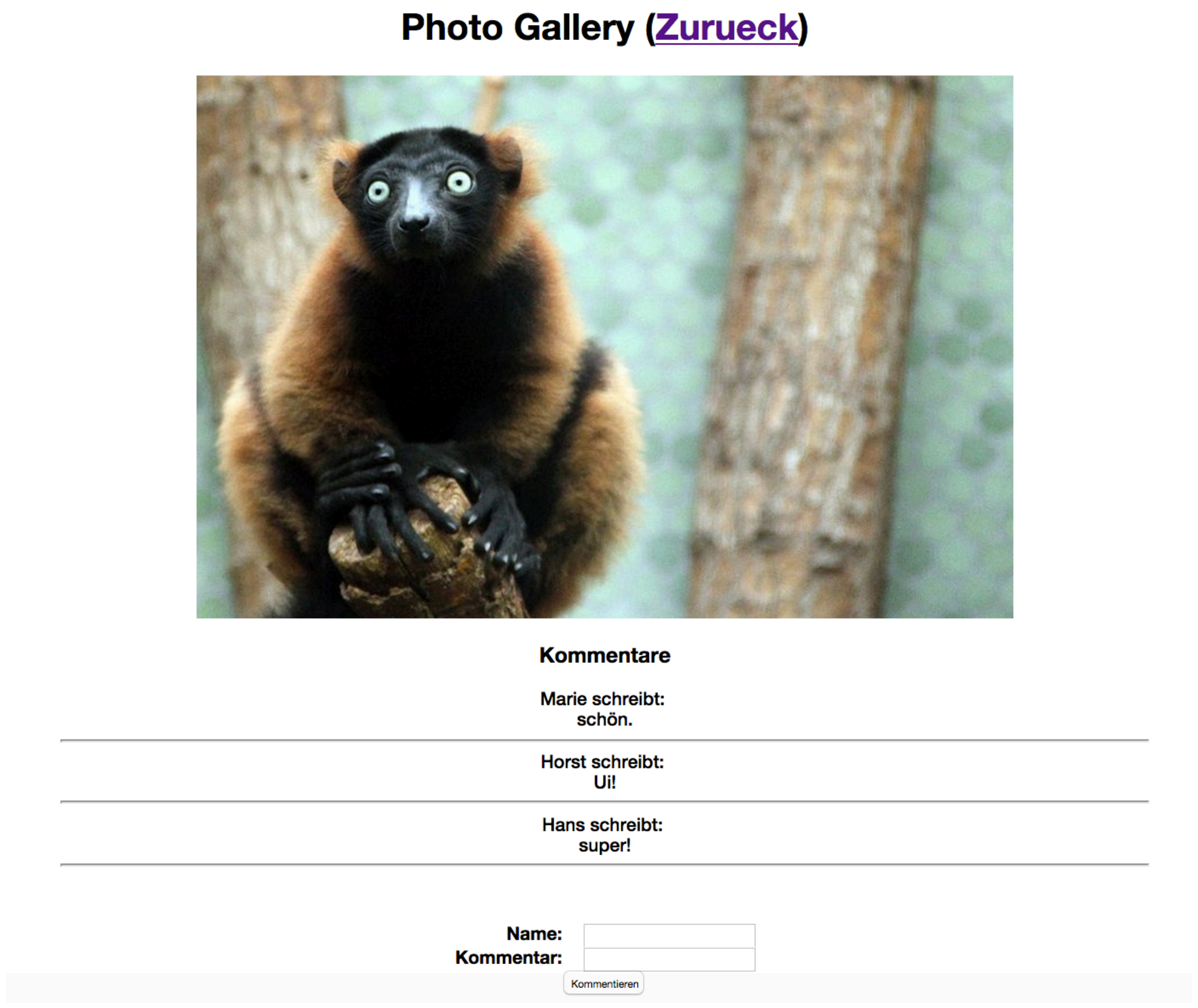


Photo Gallery (Zurueck)

Kommentare

Marie schreibt:
schön.

Horst schreibt:
Ui!

Hans schreibt:
super!

Name:	
Kommentar:	

HTML5

- HTML5 introduced a couple of new features:
  - New Elements:
    - **<canvas></canvas>**
    - <audio></audio>
    - <video></video>
    - …
  - Form features (examples):
    - Wildcards
    - Validation
    - …
  - Drag and Drop

HTML5: Document Structure

```
<!DOCTYPE html>
<html lang="de">
<head>
   <meta charset="UTF-8" />
   <title>HTML5 Structure<title>
</head>

<body>
</body>
</html>
```

HTML5: Canvas

- The <canvas> element is a container that‘s embedded into the HTML markup

```
<canvas width="400" height="400”
       style="border:1px solid #000000;">
       Browser does not support the canvas tag.
</canvas>
```

- HTML5 uses the *immediate mode* for the <canvas> element and not the *retained mode*.

HTML5: Context

- The drawing is done via JavaScript. In order to draw, the context is required: **`getContext();`**
- The context is an object that has its own attributes and methods that you can use to draw on the canvas.
- There are two types of contexts:
  - 2D
  - 3D (WebGL)

JavaScript

- JavaScript is a dynamic scripting / programming language
- Code is interpreted by the web browser
- Code can be embedded into HTML

```
<script>
<!–
   Here goes your script
-->
</script>
```

- Alternatively, the code can be imported from a file

```
<script src="myScript.js"></script>
```

DOM (Document Object Model)

- The DOM references every element and its contet in an HTML (or XML) document.
- Elements, contents and structure can be modified:
  - `document`: Content of the browser window
  - `getElementById()`: gets an HTML element with a unique identifier
  - `getElementByTagName()`: gets all elements by a specific tag
  - `Knoten.firstChild`: returns the first child node
  - `Knoten.nodeValue`: gets or sets the value of a node

- http://wiki.selfhtml.org/wiki/JavaScript
  http://de.selfhtml.org/javascript/index.htm

DOM and JavaScript

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8"/>
    <title>HTML 5</title>
</head>
<body>
    <canvas id="canvas" width="400" height="400"
           style="border:1px solid #c3c3c3;">
        Your browser does not support the HTML5 canvas tag.
    </canvas>

<script>
    var canvas=document.getElementById("canvas");
</script>
</body>
</html>
```

Retrieve the canvas‘ context

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8"/>
    <title>HTML 5</title>
</head>
<body>
    <canvas id="canvas" width="400" height="400"
           style="border:1px solid #c3c3c3;">
       Your browser does not support the HTML5 canvas tag.
    </canvas>

<script>
    var canvas=document.getElementById("canvas");
    var context = canvas.getContext("2d");
</script>
</body>
</html>
```

JavaScript and Canvas

- Colors, strokes, fills:
  - fillStyle
  - strokeStyle

- Draw rectangles
  - rect();
  - fillRect();
  - strokeRect();

- Draw images onto the canvas
  - drawImage()

- More functions:
  http://www.w3schools.com/tags/ref_canvas.asp

Draw a rectangle

```
…
<script>
    var canvas=document.getElementById("canvas");
    var context = canvas.getContext("2d");

    context.fillStyle="#00ff00";
    context.fillRect(0,0, 150, 100);
</script>
</body>
</html>
```

Exkursus: Object oriented JavaScript (I)

- The „normal“ programming style brings along a couple of disadvantages:
  - Usage of global variables
  - Variables could be overridden unintentionally
  - Including multiple JS-files can lead to conflicts
  - Loss of readability
- Idea: Combine attributes and methods into an object.

Exkursus: Object oriented JavaScript (II)

- There are different options to create objects in JavaScript:
  - Contructor functions
  - Object literal notation

- Which option should you prefer?
  - ... it depends on the problem at hand....
  - Constructors:
    - Useful if you need multiple instances of an object
  - Object literal notation:
    - If you only need one instance of an object
    - Useful for namespacing.

Example: Constructor (I)

```
function Rabbit(){
    this.adjective = "fat";
    this.whatAmI = function(){
        alert("I am a " + this.adjective + " Rabbit!");
    }
};

var fatRabbit = new Rabbit(); fatRabbit.whatAmI();
```

- Attributes are variables
- Methods are functions

Example: Constructor (II)

```javascript
function Rabbit(adjective){
    this.adjective = adjective;
    this.whatAmI = function(){
        alert("I am a " + this.adjective + " Rabbit!");
    }
};

var fatRabbit = new Rabbit("fat");
fatRabbit.whatAmI();

var whiteRabbit = new Rabbit("white");
whiteRabbit.whatAmI();
```

Example: Object Literal Notation

```
var rabbit = {
   adjective : 'fat',
   whatAmI : function(){
      alert("I am a " + this.adjective + " Rabbit!");
   }
};

rabbit.whatAmI();

rabbit.whatAmI(); rabbit.adjective = "black";
rabbit.whatAmI();
```

Example: Object Attributes

```
var myObj = {};
var obj = new Object();
var str = "myString";
var rand = Math.random();

myObj.type = "Dot syntax";
myObj["date created"] = "String with
space";
myObj[str] = "String value";
myObj[rand] = "Random Number";
myObj[obj] = "Object";
myObj[""] = "Even an empty string";
```

Closures

- Functions have their own scope in JavaScript. Inside functions you can declare:
  - Variables
  - Functions (= inner functions)
- Functions can also have **functions** as return type!
- Functions ‘remember‘ the environment in which they were created
- **Closures are special Objects that combine functions and a snapshot of an environment**
- Be careful with using `this` inside inner functions, because it could point to something else inside the closures!

Example – Nested Scopes

```
function init(){
    var testStr = "Hello!";
    function popUp(){
        alert(testStr);
    }
    popUp();
}
init();
```

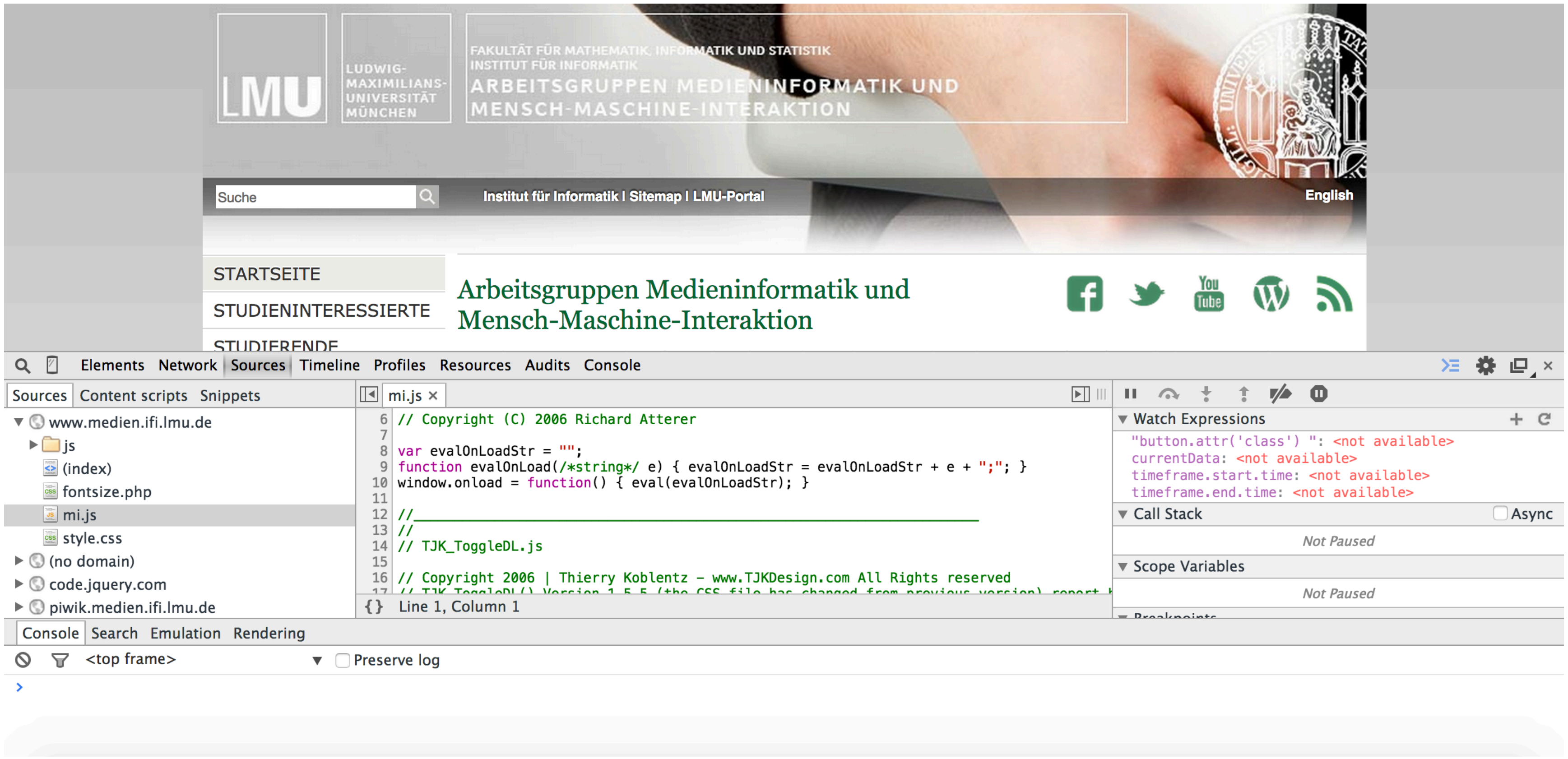
Closures – Example

```
function funky() {
    var testStr = "Hello!";

    function popUp() {
        alert(testStr);
    }

    return popUp;
}
var myFunky = funky();
myFunky();
```

Closures – Further thoughts

- Closures can become useful short links to otherwise cumbersome functions
- It is even possible to define ‘private’ methods with closures
- More information: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Closures

Debugging Javascript (I)



```
// Copyright (C) 2006 Richard Atterer

var evalOnLoadStr = "";
function evalOnLoad(/*string*/ e) { evalOnLoadStr = evalOnLoadStr + e + ";"; }
window.onload = function() { eval(evalOnLoadStr); }

//___________________________________________________________________
//
// TJK_ToggleDL.js

// Copyright 2006 | Thierry Koblentz - www.TJKDesign.com All Rights reserved
```

Line 1, Column 1

```
"button.attr('class') ": <not available>
currentData: <not available>
timeframe.start.time: <not available>
timeframe.end.time: <not available>
```

Debugging Javascript (II)

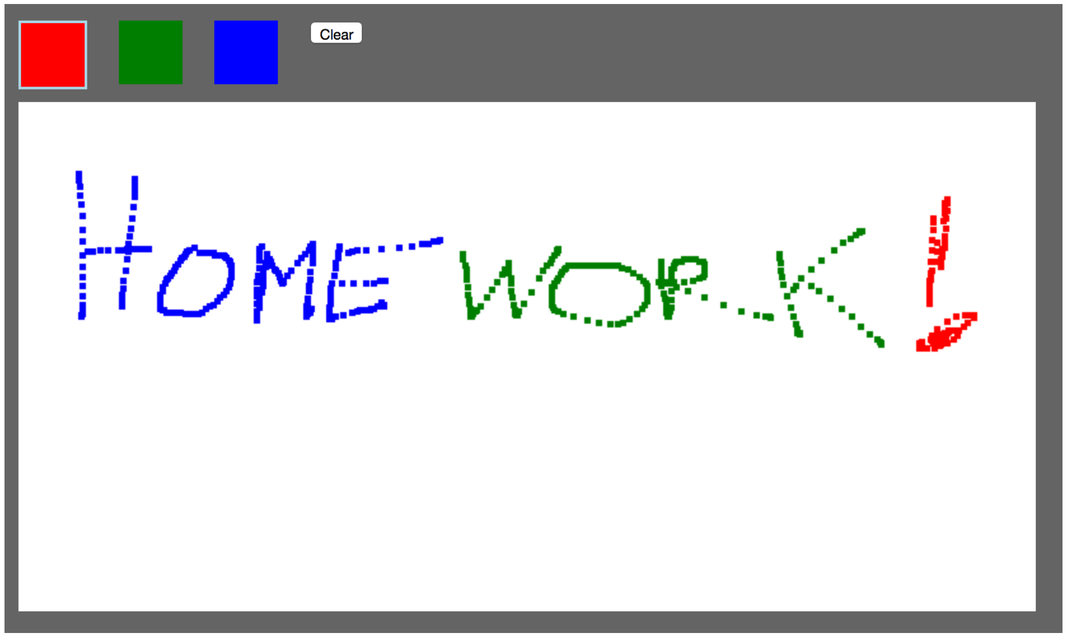
```
var check = {
    one : "Chk",
    two : "Chk",
    done : "CheckDone"
};

console.log(check);
```

Helpful Editors and IDEs

- IntelliJ WebStorm
  Free for students!
- Sublime Text
- Open Source:
  - Aptana
  - Komodo

Assignment 3

- **Topic: Drawing in the Browser**
- Due in: 1 Week
- Due date: 03.11.2014 14:00 Uhr



Thanks!

What are your questions