

Letzte Vorlesung

Themen der letzten Vorlesung:

- XSL: Templates, Anweisungen
- Beispiele für XSL-Transformationen
- Clientseitige Webprogrammierung
- JavaScript
- Objekt-orientiertes Programmieren in JavaScript

Objekt definieren und instanziiieren

```
function Text(contents){  
    this.contents = contents;  
}  
  
bla = new Text("Brabbel");  
  
document.write(bla.contents);
```

Methoden hinzufügen

Methoden sind Instanzvariablen, die eine Funktion als Wert haben:

```
function Text(contents){  
    this.contents = contents;  
}
```

```
function TextWrite() {  
    document.write(this.contents);  
}
```

```
bla = new Text("Brabbel");  
bla.write = TextWrite();
```

```
bla.write();
```

Moment! Was ist `this`, wenn `TextWrite()` anstatt `bla.write()` aufgerufen wird?

Globales Objekt

Es gibt ein vordefiniertes *globales Objekt*

- Alle global definierten Funktionen und Variablen sind Felder des globalen Objektes
- `this` zeigt auf das globale Objekt, es sei denn ein Methodenaufruf wird ausgewertet

Beispiel: globales Objekt

```
js> this;
[object global]
js> brand = "Lada";
Lada
js> this.brand;
Lada
js> function getBrand() { return this.brand; }
js> function Car(b) { this.brand = b; }
js> var benz = new Car("Benz");
js> benz.getBrand = getBrand;
```

```
function getBrand() {
    return this.brand;
}
```

```
js> benz.getBrand();
Benz
js> getBrand();
Lada
```

Nochmal: Methoden hinzufügen

Ein eleganterer Weg Methoden hinzuzufügen:

```
function Text(contents) {  
  
    this.contents = contents;  
    this.write = TextWrite;  
  
    function TextWrite() {  
        document.write(this.contents);  
    }  
}
```

- Die Funktion `TextWrite` ist nur im Scope von `Text` sichtbar
- `TextWrite` kann nur als Methode aufgerufen werden

Geheimnisse...

Was ich nicht über Funktionen erzählt habe:

Funktionen, sind eigentlich auch Objekte und haben Felder und Methoden.

Noch ein Geheimnis:

Neue Objekte entstehen durch das Kopieren bestehender Objekte.

Prototypen

Das Feld `prototype` eines Funktions-Objektes, bestimmt das Objekt, das kopiert werden soll, wenn die Funktion als Konstruktor verwendet wird.

Was passiert also bei `var obj = new Cons();`?

- Das Objekt `Cons.prototype` wird kopiert (evt. leeres Objekt)
- `this` wird auf die Kopie des Objektes gesetzt
- Die Funktion `Cons` wird ausgewertet

Prototypen und Vererbung

Durch

```
SubCons.prototype = new SuperCons()
```

wird ein `SuperCons`-Objekt zum Prototypen, der von `SubCons` erzeugten Objekte gemacht

Vererbung (1)

```
function Point(x, y) {  
  this.x = x;  
  this.y = y;  
}
```

```
function Shape(p) {  
  
  this.origin = p;  
  this.getDistance = ShapeGetDistance;  
  
  function ShapeGetDistance() {  
    return Math.sqrt((this.origin.x * this.origin.x) +  
                     (this.origin.y * this.origin.y));  
  }  
}
```

Vererbung (2)

```
function Circle(p, radius) {  
  this.origin = p;  
  this.radius = radius;  
}
```

```
Circle.prototype = new Shape();
```

```
var c = new Circle(new Point(12, 14), 20);
```

```
c.getDistance();  
c.origin.x;
```

Arrays

Arrays werden mit vordefiniertem Objekt erzeugt.

- Erzeuge mit `new Array(length)`
- Oder: `new Array(element-1, element-2)`

Referenzieren und Modifizieren mit `[]`

Es gibt keine Bereichsüberprüfung:

- Referenzieren außerhalb liefert `undefined`
- Modifizieren außerhalb fügt Element hinzu

Index kann ein beliebiges Objekt sein

AJAX

AJAX steht für Asynchronous JavaScript And XML

Idee:

- JavaScript verändert die Baumrepräsentation einer XHTML-Seite oder tauscht Inhalte aus
- Die Seite wird also verändert, statt komplett neu geladen
- JavaScript-Code lädt bei Bedarf (kleinere Mengen) Informationen als XML nach

⇒ Kürzere Wartezeiten und flüssigere Bedienung

Voraussetzungen

Voraussetzungen für AJAX-Applikationen:

- Zugriff auf die Baumrepräsentation der XHTML-Seite. Ideal: Programmierschnittstelle ist browserunabhängig
- Funktionen, um XML-Dokumente per HTTP anzufordern und zu parsen

Document Object Model

Document Object Model (DOM)

- Beschreibt, wie HTML- und XML-Dokumente in objekt-orientierten Programmiersprachen repräsentiert werden
- Programmierschnittstelle für Zugriff auf die Repräsentation
- Entspricht ungefähr der vorgestellten Baumrepräsentation für XML

W3C: Serie von sprach- und plattformunabhängigen Spezifikationen für DOM

In dieser Vorlesung: Nur JavaScript-API für DOM

Involvierte Objekte

Zur Repräsentation eines Dokumentes werden diese Objekte verwendet:

- `Attr`, `Element`, `Entity`, `Text`
- `CDATASection`, `CharacterData`, `Comment`
- `Document`
- `NodeList`
- `ProcessingInstruction`
- ...

Fast alle Objekte erben von `Node`

Node-Objekte

Ein `Node`-Objekt repräsentiert einen Knoten in der DOM-Repräsentation eines Dokumentes

- `childNodes: NodeList`
Ein Liste aller Knoten, die unterhalb des Knotens liegen
- `parentNode: Node`
Elternknoten des Knotens
- `appendChild(child), removeChild(child) -> Node`
Füge einen Kindknoten hinzu, bzw. löscht Kindknoten

document-Objekte (1)

Die `document`-Objekte repräsentieren HTML- oder XML-Dokumente.

Vordefinierte Variable: `document` repräsentiert das momentan dargestellte Dokument

Wichtige Methoden, um Knoten zu erzeugen

- `createElement(tagname) -> Element`
Neues Element erzeugen (aber nicht einfügen!)
- `createTextNode(string) -> Text`
Neuen Textknoten erzeugen (aber nicht einfügen!)
- `createAttribute(attributeName) -> Attr`
Neues Attribut erzeugen (aber nicht einfügen!)

document-Objekte (2)

Methoden, um Elemente zu suchen:

- `getElementById(string) -> Element, null`
Findet ein Element über das id-Attribut
- `getElementsByTagName(tagName) -> NodeList`
Liste aller Elemente des gegebenen Namens

Andere Methoden:

- `importNode(node, deep) -> Node`
Kopiert Knoten von einem Dokument in ein anderes

Element-Objekte

Element-Objekte repräsentieren ein Element-Knoten, bestehend aus Tag, Attributen und Inhalt

- `hasAttribute(name) -> Boolean`
Überprüfen, ob Attribut name vorhanden
- `getAttribute(name) -> String`
Wert des Attribute name zurückgeben
- `getElementsByTagName(name) -> NodeList`
Liste der Kind-Elemente vom Typ name
- `childNodes: NodeList`
Geerbt von Node: Liste aller Kind-Knoten

NodeList-Objekte

NodeList-Objekte repräsentieren eine Menge von Knoten.

- **length: Number**
Anzahl der Knoten in der Liste
- **item(index) -> Node**
Knoten an Position index zurückgeben

XMLHttpRequest

`XMLHttpRequest` lädt eine XML-Datei von einer URL und macht diese als DOM-Repräsentation zugänglich

- `open(method, url, async) -> void`
Setzt die URL für einen Request
- `send(body) -> void`
Setzt den Rumpf der HTTP-Anfrage
- `responseXML: document`
Das empfangene XML-Dokument

Asynchrone Anfragen

`XMLHttpRequest` unterstützt asynchrone Anfragen. In diesem Fall blockiert `send` nicht, sondern kehrt sofort zurück.

Status-Änderungen der Anfrage über Callback:

```
request.onreadystatechange = function () {  
    if (request.readyState == 4) {  
        if (request.status == 200) {  
            /* Request erfolgreich */  
        } else {  
            alert("Request fehlgeschlagen")  
        }  
    }  
}
```

JavaScript und Events

GUI-Events im Browser lassen sich mit einem JavaScript- Callback verknüpfen.

Tritt das Event ein, wird der Callback ausgelöst

```
<script type="text/javascript">  
function annoy_user() {  
    alert("Event eingetreten!");  
}  
</script>
```

```
<div onmouseover="annoy_user">  
...  
</div>
```

Andere Events:

`onclick ondblclick onfocus onkeydown ...`

Beispiel-Anwendung

Beispiel-Anwendung: Kalender mit Vorschau

- Zeigt Kalendarium und ein Vorschau-Feld
- Fährt man mit der Maus über einen Tag, werden die Termine im Vorschau-Feld angezeigt
- Termine werden per JavaScript nachgeladen

daydata.php (1)

```
<?php
$day = $_REQUEST['d'];
$month = $_REQUEST['m'];
?>
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <body>
    <p>
      <b>Appointments for <?= $day.'/'.$month ?></b>
    <ul>
```

daydata.php (2)

```
<?php
$count = rand(3, 10);

for ($i = 1; $i < $count; $i++)
    $appointments[rand(9, 18)] = "Appointment $i on $day $month";
ksort($appointments);

foreach ($appointments as $key => $val)
    echo "<li><i>$key:00 Uhr</i> $val</li>\n";
?>
    </ul>
</p>
</body>
</html>
```

index.php (1)

```
<script type="text/javascript">
<!--
function make_data_available_callback(request) {

return function () {
    if (request.readyState == 4) {
        if (request.status == 200) {
            var new_doc = request.responseXML;
            var paragraph_doc = new_doc.getElementsByTagName('p').item(0);
            var paragraph = document.importNode(paragraph_doc, true);
            var preview_area = document.getElementById('preview_area');
            preview_area.removeChild(preview_area.firstChild);
            preview_area.appendChild(paragraph);
        }
        else alert("Error loading page");
    } else
        return undefined;
}
}
```

index.php (2)

```
function preview(day, month, year) {  
  
    request = new XMLHttpRequest();  
    request.onreadystatechange =  
        make_data_available_callback(request);  
    request.open('GET',  
                <?= '' .dirname($_SERVER['SCRIPT_NAME']).  
                '/daydata.php?d='' ?>  
                + day + '&m=' + month,  
                true);  
    request.overrideMimeType('text/xml');  
    request.send(null);  
}  
-->  
</script>
```

index.php (3)

```
<?php

$month = isset($_REQUEST['m']) ? $_REQUEST['m'] : date('n');
$year = 2006;

function output_calendar($month, $year, $fun)
{
    $ts = mktime(0, 0, 0, $month, 1, $year);
    $num_days = date('t', $ts);
    $first_day = date('w', $ts)+1;

    echo '<table><tr style="background-color:#CCCCCC">';
    echo '<td>So</td><td>Mo</td><td>Di</td><td>Mi</td>';
    echo "&<td>Do</td><td>Fr</td><td>Sa</td></tr>\n";
    echo "<tr>\n";
```

index.php (4)

```
for ($i = 1, $j = 1; $i < 43; $i++) {
    if ($j > $num_days)
        echo '<td>&nbsp;</td>';
    else
        if ($i >= $first_day)
            echo '<td>'.$fun($j++).'</td>';
        else
            echo '<td>&nbsp;</td>';
    if ($i % 7 == 0)
        echo '</tr><tr>';
}

echo '</tr>';
echo '</table>';
}
```

index.php (5)

```
function add_on_mouseover_event($day) {
    global $year, $month;
    return "<div onmouseover=\"preview($day, $month, $year);\">".
        $day."</div>";
}
?>
```

```
<table>
<tr align="left" valign="top">
<td>
    <?php output_calendar($month, $year,
        'add_on_mouseover_event'); ?>
</td>
<td style="background-color:#FFFFCC"
    width="300px" height="400px">
<p><b>Preview area</b><br>
    <div id="preview_area"/>
</p>
</td></tr>
</table>
```