Webseitengestaltung mit JavaScript

JavaScript ist kein direkter Bestandteil von **HTML** sondern eine eigene Programmiersprache. Diese Sprache wurde jedoch eigens zu dem Zweck geschaffen, HTML-Autoren ein Werkzeug in die Hand zu geben, mit dessen Hilfe sich Webseiten optimieren lassen. JavaScripts werden wahlweise direkt in der HTML-Datei oder in separaten Dateien notiert. Sie werden zur Laufzeit vom Webbrowser interpretiert. Dazu besitzen moderne Webbrowser entsprechende Interpreter-Software. JavaScript wurde 1995 von Netscape eingeführt und lizenziert. Netscape diktierte also, woraus das "offizielle" JavaScript bestand. Mittlerweile ist die Situation aus verschiedenen Gründen unübersichtlicher geworden…

Das **Document Object Model (DOM)** ist eine vom W3-Konsortium verabschiedete Norm, die zunächst einmal den Zugriff einer Programmiersprache auf beliebige Elemente eines Auszeichnungssprachen-Dokuments beschreibt. Das DOM ist also weder selber eine eigene Programmiersprache, noch ist es auf HTML beschränkt. Es definiert lediglich Objekte, Eigenschaften und Methoden, die eine Programmiersprache umsetzen sollte, wenn sie sich als DOM-fähig bezeichnen will. Anwendbar sollen diese Objekte, Eigenschaften und Methoden auf alle Dokumente sein, die in einer XML-gerechten Auszeichnungssprache geschrieben sind. JavaScript entspricht dieser DOM-Norm.

Wir lernen also das Browserfenster, das geladene Dokument und weiteres als Objekte neu kennen. Zu jedem Objekt gibt es dann eine Reihe von:

- Eigenschaften,
- Methoden (die „Verben" der Websitegestaltung),
- Unterobjekten.

Dynamisches HTML (engl. "Dynamic HTML" oder abgekürzt "**DHTML**") ist eine Erfindung von Marktstrategen, sagen Kritiker. In der Tat ist Dynamisches HTML keine klassische HTML-Erweiterung in Gestalt neuer HTML-Elemente. Es ist auch keine neue Sprache. Dynamisches HTML ist vielmehr der Sammelbegriff für verschiedene Lösungen, um dem Autor einer Web-Seite zu ermöglichen, Elemente der Webseite während der Anzeige dynamisch zu ändern, sei es automatisch oder durch Einwirken des Anwenders.

Java ist eine 1995 von der Firma Sun entwickelte, Plattform (Betriebssystem) unabhängige Programmiersprache und wird daher häufig auch in Webseiten eingesetzt, um animierte oder interaktive Inhalte (wie z.B. Spiele) zu erzeugen. Java-Programme, die zur Anzeige innerhalb des Browserfensters erstellt wurden, werden als Java-Applets bezeichnet. Die Programmierung mit Java ist komplex und sprengt den Rahmen dieses Kurses. In kompilierter Form mit der Erweiterung *.class werden sie in die HTML-Datei durch das Tag <object> (früher auch <applet>) eingebunden. Das Entwicklerkitt ist für private Zwecke kostenfrei.

Das von Macromedia entwickelte und von Adobe fortgeführte **Flash** bietet die Möglichkeit, Multimedia-Effekte auf Web-Seiten zu bringen, aber auch Anwendungen wie Spiele, Simulationen oder Navigations-Unterstützung für Web-Seiten. Es gilt als sehr modern. Da hinter Flash ein kommerzielles Software-Produkt steckt, das in keiner Weise mit den offenen Internet-Standards vergleichbar ist, werden also Äpfel mit Birnen verglichen. Flash steht vor allem dort hoch im Kurs, wo man der Meinung ist, auf einer Web-Seite müsse es zugehen wie im Fernsehen. Manche Kreativlinge unterliegen diesem Glauben, aber noch mehr die Marketing-Abteilungen von Firmen, die bei Agenturen ihren Web-Auftritt gestalten lassen und dafür am Bildschirm etwas sehen wollen, von dem sie glauben, das sei besonders schwierig und atemberaubend. Alles in allem ist Flash also ein mächtiges (und kostenpflichtiges) Werkzeug, um Inhalte zu visualisieren. Auch Flash muss in eine erste HTML-Seite eingebettet werden. Die Möglichkeiten von Suchmaschinen, Flashseiten zu indizieren, sind begrenzt.

Nicht unerwähnt bleiben sollte jedoch, dass es auch frei verfügbare, offen dokumentierte und letztlich noch leistungsfähigere Internet-Standards für das gibt, was Flash leistet. So werden beim W3-Konsortium zwei **XML**-basierte Sprachen namens **SVG** und **SMIL** standardisiert, die in Verbindung mit JavaScript und DOM ähnliche Möglichkeiten eröffnen.

Heute ist außerdem viel von **AJAX** die Rede. Der Name ist ein Apronym für „**A**synchronous **J**avaScript **a**nd **X**ML". JavaScript ist eine der Grundlagen für diesen Standard. Bekannte Webprojekte basieren auf AJAX. Zu nennen wären: Google Maps, Flickr, Last.fm und Facebook.

Literatur-Tipps

Lehrgänge zum Selbststudium

HTML 4 für Dummies
Ed Tittel & Natanya Pitts
mitp-Verlag
ca. 25,00 €

JavaScript für Dummies
Emily A. Vander Veer
mitp-Verlag
ca. 25,00 €

Web-Academy (X)HTML
Mechthild Käufer
Data Becker
ca. 25,00 €

Web-Academy JavaScript
Antje Hofmann
Data Becker
ca. 25,00 €

Kurzreferenzen (Taschenbücher, Online)

HTML – kurz & gut
Jennifer Niederst

CSS – kurz & gut
Eric A. Meyer

JavaScript – kurz & gut
David Flanagan

Alle drei bei: O'Reilly Verlag
jeweils um die 10,00 €

SelfHTML Version 8.1.2
SelfHTML e.V. (ehemals Stefan Münz)
Kostenlos unter http://de.selfhtml.org/

Script-Bibliotheken (Online)

Andre's JavaScripte
Andre Göntgen
http://www.javascripte.org/

brauchbar.de
Thomas Salvador
http://www.brauchbar.de/

HTML-World
Jan Winkler
http://www.html-world.de/

Web-Toolbox
Wilhelm Jansen
http://www.web-toolbox.net/

JRP Webdesign
Johann Piber
http://www.jrp.at/beispiele/

jswelt.de
Michael Ohnesorg & Tobias Bohlinger
http://www.jswelt.de/

Michael Mailer's JavaScripte
Michael Mailer
http://www.mywebaid.de/

Kostenlose Javascripts (mit Werbung)
Michael Antrag
http://www.kostenlose-javascripts.de/

Allgemeine Informationen zum Computer und zum Internet

Duden – Basiswissen Schule Angewandte Informatik
Dr. Lutz Engelmann (Hrsg.)
Dudenverlag
21,00 €

Taschenbuch Multimedia
Peter A. Henning
Fachbuchverlag Leipzig
ca. 20,00 €

JavaScript

Bevor Sie JavaScript-Code programmieren, müssen Sie sich exakt darüber im klaren sein, welches Problem Sie damit lösen wollen, was man mit HTML selbst alles machen kann und wo die Grenzen von HTML liegen. Von JavaScript müssen Sie dann so viel wissen, dass Sie entscheiden können, ob das Problem mit JavaScript überhaupt lösbar ist. JavaScripts werden im Browser des Anwenders ausgeführt, nicht auf dem Server, wo die Webseiten abgelegt sind. Das heißt, JavaScript kann vom Browser erst dann ausgeführt werden, wenn er eine Webseite gerade einliest oder nachdem er sie eingelesen hat – dann auch ausgelöst durch Benutzerereignisse wie z.B. Mausklicks.

Aus diesem Grund können Sie mit JavaScript also keine Anwendungen wie Gästebücher oder Web-Foren programmieren. Denn solche Anwendungen müssen die Daten aller beitragenden Anwender zentral auf dem Server speichern. Wohl aber kann JavaScript unterstützend eingesetzt werden. So ist es beispielsweise sinnvoller, Formulareingaben des Anwenders vor dem Absenden mit Hilfe von JavaScript zu überprüfen, als den Server-Rechner mit dieser Arbeit zusätzlich zu belasten.

Es lohnt sich durchaus, im Netz nach frei verfügbaren JavaScript-Beispielen zu suchen, die genau Ihr Problem lösen. Denn auf Code zurückzugreifen, der sich im Einsatz bereits bewährt hat, erspart Ihnen viel Ausprobieren. In vielen Fällen genügt es, vorhandene JavaScripts den eigenen Erfordernissen anzupassen. Bei vorhandenen JavaScripts müssen Sie allerdings so viel von der Sprache verstehen, dass Sie genau wissen, welche Variablen oder festen Werte Sie verändern oder welche Funktion Sie ergänzen wollen.

Die Einbettung von JavaScript in ein HTML-Dokument

```
<head>
…
<script type="text/javascript">
<!--
      JavaScript Anweisungen
      …
//-->
</script>
…
</head>
```

als Inhalt des Script-Tags, umschlossen von Kommentarzeichen

```
<head>
…
<script type="text/javascript"
      src="url">
</script>
…
</head>
```

als Link auf ein externes JavaScript-Dokument

Grundlagen der JavaScript-Syntax

JavaScript ist objektorientiert. Objekte sind die Einheiten, denen Eigenschaften, Methoden und Unterobjekte zugeordnet sind. Methoden sind Funktionen, die Aktionen ausführen, aber im Gegensatz zu alleinstehenden Funktionen an ein bestimmtes Objekt gebunden sind. Viele vordefinierte JavaScript-Objekte haben Methoden.

JavaScript besteht letztendlich aus einer kontrollierten Anordnung von Anweisungen. Das sind Befehle, die der JavaScript-Interpreter des Browsers bewertet und in Maschinencode umsetzt, der auf dem Rechner des Anwenders ausführbar ist. Eine Anweisung in JavaScript besteht immer aus einem Befehl, der mit einem Semikolon [**;**] oder einem Zeilenumbruch abgeschlossen wird. Zur Sicherheit sollten Sie sich beides angewöhnen. Ein Anweisungsblock wird durch eine geschweifte Klammer `{` begonnen und durch eine geschweifte Klammer `}` beendet.

Eine Anweisung ist zum Beispiel:

- wenn Sie einer Variablen einen Wert zuweisen [`Zahl = 42;`]
- wenn Sie eine Operation durchführen [`Ergebnis = Zahl * Zahl;`]
- wenn Sie einen bedingten Befehl ausführen [`if(Zahl > 1000) Zahl = 0;`]
- wenn Sie eine Funktion oder eine Methode aufrufen, bzw. wenn Sie eine Objekteigenschaft auslesen oder ändern [`alert("Das Quadrat von " + Zahl + " = " + Ergebnis);`]

Wenn Sie Ihren JavaScript-Code kommentieren wollen, um bei einer späteren Bearbeitung gleich zu wissen, wofür Sie das Script geschrieben haben, können Sie dies im Script selbst tun:

```
// Ein einzeiliger Kommentar.
/* Ein mehrzeiliger Kommentar */
```

Achten Sie auf Groß- und Kleinschreibung. JavaScript ist casesensitive. Manche Wörter oder einzelne Zeichen sind bei JavaScript verboten, da sie mit spezifischen Bedeutungen belegt sind. Strings werden durch doppelte oder einfache Anführungsstriche gekennzeichnet. Wenn nun im String Anführungsstriche benutzt werden sollen, müssen sie zuerst durch ein Backslash entwertet oder durch einfache/doppelte ersetzt werden.

```
”String mit ‘Wort‘ in Anführungzeichen”
”String mit \”Wort\” in Anführungzeichen”
```

`\\` = ein einfacher Backslash als String
`\n` = ein Zeilenumbruch im String

Das Objekt `window` (Fenster) ist das oberste Objekt der Objektfamilie für alles, was im Browser-Fenster angezeigt wird. Das aktuelle Fenster des Browsers können Sie über die Objekte `window` oder `self` ansprechen. Mit der folgenden Anweisung wird die Anzeige in der Statusleiste bestimmt.

```
window.defaultStatus = "Meine Homepage";
```

Die nächste Anweisung fragt nach dem Namen des Objekts `navigator` und gibt diesen in Verbindung mit einem String in die geladene HTML-Datei (das Objekt `document`) aus.

```
document.write("So, so, Sie verwenden also " + navigator.appName);
```

Die erste Funktion

Nun folgt schon die erste selbst geschriebene Funktion mit dem Namen `NeuFenster`. Es wird die Variabel `MeinFenster` definiert, die aus dem Objekt `window` mit der Methode `open` und einigen Eigenschaften besteht. Schließlich wird das Objekt `MeinFenster` mit der Methode `focus` in den Vordergrund gestellt oder aktiviert.

```
function NeuFenster() {
MeinFenster = window.open("datei2.html", "TargetName",
"width=300,height=200,scrollbars");
MeinFenster.focus();
}
```

Wenn nun im Body der HTML-Datei folgender Link eingesetzt wird, öffnet dieser beim Klick ein neues Fenster mit vorbestimmten Eigenschaften.

```
<a href="javascript:NeuFenster()">Hier klicken für neues Fenster</a>
```

Mit diesem Link kann das Hauptfenster das neu geöffnete Fenster auch wieder schließen.

```
<a href="javascript:MeinFenster.close()">Fenster schließen</a>
```

Im HTML-Dokument des zweiten Fensters könnte der nächste Link untergebracht sein, der das Hauptfenster, den `opener`, schließt. Aus Sicherheitsgründen würde ein moderner Browser das Hauptfenster nicht kommentarlos schließen, sondern mit einem Dialogfenster noch einmal nachfragen.

```
<a href="javascript:opener.close()">Hauptfenster schließen </a>
```

Mit dieser zweiten Funktion lässt sich nun prüfen, ob das Zweitfenster geschlossen wurde. Je nach dem wird ein Dialogfenster mit einem Hinweis ausgegeben.

```
function CheckOpen() {
if(MeinFenster.closed == true) alert("Fenster wurde geschlossen");
else alert("Fenster noch offen");
}
```

Anweisungen

Von den JavaScript-Anweisungen möchte ich hier sechs kurz vorstellen. Einige haben wir in den Beispielen weiter oben schon verwendet. Lassen Sie sich bitte nicht von der doppelten Verwendung des Begriffs Anweisung irritieren.

`function` Sie definiert eine Funktion, z.B. diese namens Funktionsname, deren Body aus einer oder mehr Anweisungen besteht und die mit keiner oder mehr mit Komma voneinander getrennten Argumenten versehen ist.

```
function Funktionsname (Argumente) {
Anweisungen; }
```

`if/else` Sie führt eine Anweisung aus, wenn ein Ausdruck wahr ist, sonst führt sie eine andere Anweisung aus.

```
if (Ausdruck) {Anweisungen1; }
[else {Anweisungen2; }]
```

`return` Sie beendet die Ausführung der laufenden Funktion und kehrt zu ihrem Aufrufer zurück. Wenn ein Ausdruck folgt, wird dessen Wert als Rückgabe der Funktion verwendet.

```
return [Ausdruck] ;
```

`switch/case` Sie prüft, ob ein berechneter Ausdruck einem bestimmten konstanten Wert entspricht und führt die zugeordneten Anweisungen aus. Gibt es keine Entsprechung, werden die default Anweisungen ausgeführt.

```
switch (Ausdruck) {
case konstanterWer1: Anweisungen1;
break;
[case konstanterWert2: Anweisungen2;
break; ]
default: Anweisungen; }
```

`var` Sie deklariert und initialisiert (optional) eine oder mehr Variabeln.

```
var Name [= Wert] [, Name2 = Wert2]
```

`while` Sie ist eine elementare Schleife. Sie führt eine Anweisung aus, so lange ein Ausdruck wahr ist.

```
while (Ausdruck) {Anweisung; }
```

Objekte und ihre Methoden

Die Methoden sind Aktionen, die ein bestimmtes Objekt ausführen kann. Die Auswahl möglicher Methoden ist demnach abhängig vom jeweiligen Objekt. In der Folge stelle ich Ihnen nochmals drei Objekte und eine Auswahl der möglichen Methoden vor

Mit dem Objekt `window` haben wir weiter oben schon gearbeitet. Lassen Sie mich an dieser Stelle einige Methoden in alphabetischer Reihenfolge benennen:

`alert()` Ruft ein Dialogfenster mit Informationen auf.

`blur()` Verlässt das Fenster und setzt es ganz nach hinten.

`close()` Schließt das Fenster.

`confirm()` Ruft ein Dialogfenster zur Bestätigung auf.

`focus()` Aktiviert das Fenster und setzt es somit nach vorn.

`open()` Öffnet ein neues Fenster.

`print()` Druckt den Fensterinhalt aus.

`prompt()` Ruft ein Dialogfenster zur Werteingabe auf.

Über das Objekt `history`, das in der JavaScript-Objekthierarchie unterhalb des `window`-Objekts liegt, haben Sie Zugriff auf die besuchten Web-Seiten des Anwenders. Maßgeblich ist dabei die Liste, wie sie in der History-Liste des Browsers gespeichert ist. Mit diesem Objekt können Sie z.B. Verweise vom Typ "springe zur zuletzt besuchten Seite" konstruieren.

```
<a href="javascript:history.back()">zur&uuml;ck</a>
<a href="javascript:history.forward()">wieder vor</a>
<a href="javascript:history.go(-3)">3 Seiten zur&uuml;ck</a>
```

Ein anderes Unterobjekt ist das Objekt `document`. Von diesem Objekt aus haben wir Zugriff auf die Unterobjekte aus den einzelnen HTML-Elementen und Formularen.

`close()` Schließt das Dokument. Die Bearbeitung durch ein Skript ist abgeschlossen.

`createAttribute()`
Erzeugt ein Attribut, welches in HTML-Elementen verwendet werden kann.

`getElementById()`
Greift auf HTML-Element zu über das id-Attribut.

`getElementsByName()`
Greift auf HTML-Element zu über das name-Attribut.

`write()` Schreibt ins Dokumentfenster.

Beispiel eines komplexeren Javascripts mit HTML-Formular. Mit dem `Math` Objekt lassen sich in JavaScript auch anspruchsvollere Probleme berechnen.

Im `head` des Dokuments:

```
<script type="text/javascript">
function berechne(betrag,wechselkurs) {
if (wechselkurs == "USD") ergebnis(betrag*0.6462,wechselkurs );
if (wechselkurs == "EUR") ergebnis(betrag/0.6462,wechselkurs ); }
function ergebnis(wert,wechselkurs) {
if (wechselkurs == "USD") wert=Math.round(wert*100)/100 +' Euro';
else wert=Math.round(wert*100)/100 +' US Dollar';
document.form.ausgabe.value=wert; }
</script>
```

Im `body` des Dokuments:

```
<form name="form">
Betrag eingeben: 
<input name="eingabe" size="10"> 
      <select name="waehrung" size="1">
      <option value="USD">US Dollar</option>
      <option selected value="EUR">Euro</option>
      </select>
<br><br>  
<input
onClick="javascript:berechne(eingabe.value,waehrung.options[waehrung.
selectedIndex].value)" type="button" value="Berechnen"><br><br>
Geld gewechselt: 
      <input name="ausgabe" size="23">
</form>
```

Event-Handler

Event-Handler (Ereignis-Behandler) sind ein wichtiges Bindeglied zwischen HTML und JavaScript. Event-Handler werden meist in Form von Attributen in HTML-Tags notiert. Da es sich um Bestandteile handelt, die innerhalb von HTML vorkommen, hat das W3-Konsortium die Event-Handler mittlerweile auch in den HTML-Sprachstandard mit aufgenommen.

Event-Handler erkennen Sie daran, dass solche HTML-Attribute immer mit `on` beginnen, zum Beispiel `onClick=`. Hinter dem Gleichheitszeichen notieren Sie in Anführungszeichen, eine JavaScript-Anweisung. Wenn Sie mehrere Anweisungen ausführen wollen, dann definieren Sie sich dazu am besten in einem JavaScript-Bereich eine Funktion und rufen hinter dem Gleichheitszeichen diese Funktion auf, also z.B. `onClick="Funktion()"`.

`onAbort`	(bei Abbruch)
`onBlur`	(beim Verlassen)
`onChange`	(bei erfolgter Änderung)
`onClick`	(beim Anklicken)
`onDblClick`	(bei doppeltem Anklicken)
`onError`	(im Fehlerfall)
`onFocus`	(beim Aktivieren)
`onKeydown`	(bei gedrückter Taste)
`onKeypress`	(bei gedrückt gehaltener Taste)
`onKeyup`	(bei losgelassener Taste)
`onLoad`	(beim Laden einer Datei)
`onMousedown`	(bei gedrückter Maustaste)
`onMousemove`	(bei weiterbewegter Maus)
`onMouseout`	(beim Verlassen des Elements mit der Maus)
`onMouseover`	(beim Überfahren des Elements mit der Maus)
`onMouseUp`	(bei losgelassener Maustaste)
`onReset`	(beim Zurücksetzen des Formulars)
`onSelect`	(beim Selektieren von Text)
`onSubmit`	(beim Absenden des Formulars)
`onUnload`	(beim Verlassen der Datei)
`javascript:`	(bei Verweisen)

Als Beispiel für die Verwendung von einem Event-Handler in einem HTML-Dokument sehen Sie hier den bekannten Rollover-Effekt, der Wechsel des Bildes sobald sich die Maus darüberbewegt:

```
<img src="bild1.jpg" onMouseover="this.src='bild2.jpg'"
onMouseout="this.src='bild1.jpg'">
```

Für diesen Effekt müssen Sie nicht JavaScript im Head-Bereich einbinden. Es ist aber ratsam, das zweite Bild mit folgendem Befehl vorzuladen, damit es ohne Verzögerung angezeigt werden kann.

```
<script type="text/javascript">
var a = new Image();
a.src="bild2.jpg";
</script>
```

Ein weiteres Beispiel ist die Änderung Statuszeilen-Anzeige bei einem Link:

```
<a href="help.html" onMouseover="window.status='Hier erhalten Sie
Hilfe'; return true;">Brauchen Sie Hilfe?</a>
```

Übersicht der Operatoren in JavaScript

Operation	*Symbol*	*Bedeutung*	*Beispiel*	*Wert*
Die mathematischen Operatoren				
Addition	+	*Gibt die Summe zweier Zahlen zurück*	`alert(8+5)`	13
Subtraktion	-	*Gibt die Differenz zweier Zahlen zurück*	`alert(8-5)`	3
Multiplikation	*	*Gibt das Produkt zweier Zahlen zurück*	`alert(8*5)`	40
Division	/	*Gibt den Quotienten zweier Zahlen zurück*	`alert(8/5)`	1.6
Modulo	%	*Ermittelt den Rest der ganzzahligen Division zweier Zahlen*	`alert(8%5)`	3
Inkrement	++	*Erhöht eine in einer Variablen gespeicherten Zahl um 1*	`X=4; x++; alert(x)`	5
Dekrement	--	*Vermindert eine in einer Variablen gespeicherten Zahl um 1*	`X=4; x--; alert(x)`	3
Die Vergleichsoperatoren				
gleich	==	*Stimmen beide Werte überein?*	`alert(8==5)`	false
ungleich	!=	*Sind beide Werte unterschiedlich?*	`alert ("Wut"!="Mut")`	true
kleiner/gleich	<=	*Ist der erste Wert kleiner oder gleich dem zweiten Wert?*	`alert(8<=5)`	false
kleiner als	<	*Ist der erste Wert kleiner als der zweite Wert?*	`alert ("Tag"<"S")`	false
größer/gleich	>=	*Ist der erste Wert größer oder gleich dem zweiten Wert?*	`alert(8>=5)`	true
größer als	>	*Ist der erste Wert größer als der zweite Wert?*	`alert ("Wa">"Wau")`	false
Die logischen Operatoren				
UND	&&	*Stimme beide Vergleiche überein?*	`alert(8>5 && "a"=="z")`	false
ODER	||	*Stimmt mindestens einer der Vergleiche überein?*	`alert(8<5 || 8!=5)`	true
NICHT	!	*Umkehr des Wahrheitswertes*	`alert(!(8==5))`	true

Wichtig:

- Um sich von den Operatoren nicht einschüchtern zu lassen, ist es sinnvoll, sich zuerst den zu untersuchenden Sachverhalt auf einem Papier mit eigenen Worten zu notieren.
- Die Methode `alert()` ist nur ein Beispiel. Die Operatoren finden bei sämtlichen Anweisungen und Methoden Anwendung.
- Für Dezimalzahlen wird in JavaScript immer amerikanisch der Punkt verwendet (1.6) nicht das Komma (1,6).
- Bei Vergleichen von Strings geht es zuerst nach dem Alphabet dann nach Zeichenlänge.
- JavaScript arbeitet immer von oben nach unten, von links nach rechts. Ist beim Operator UND z.B. der erste Vergleich schon falsch, wird der zweite gar nicht mehr überprüft.