assono

JavaScript leicht gemacht

Schwentinental, 01. März 2010

Innovative Software-Lösungen.

www.assono.de

assono

Manuel Nientit

Diplom-Wirtschaftsinformatik, Universität Paderborn

seit 2003 entwickle ich mit und für Lotus Notes (6.5 ff.)

seit 2009 auch Webentwicklung

seit 2008 IT-Consultant bei **assono**


mnientit@assono.de


http://www.assono.de/blog


04307/900-406

assono
IT-Consulting & Solutions

assono

Was ist JavaScript?

- JavaScript hat nichts mit Java zu tun
- Ähnlichkeiten zur Java-Syntax gegeben
- JavaScript ist eine interpretierte Sprache
- Läuft in einer Sandbox
- Zugriff auf OS-Funktionen wird noch diskutiert bzw. eingeschränkt über ActiveX
- **Lässt sich im Browser abstellen!**

Entwicklung

- Kein spezieller Editor notwendig
- Aber es gibt Hilfen:
  - Komodo Edit
  - TopStyle
  - Aptana Studio
  - Notes kann auch ;)
  - ...
- Debugging schwierig -> z.B. Firebug
- **Achtung! Es gibt Unterschiede zwischen Browsern!**

Hello World

- JS Anweisungen werden beim Laden ausgeführt
- Syntaxregeln:
  - Strings in Anführungszeichen - auch einfache
  - Jede Anweisung wird mit ";" beendet

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Hello World</title>
        <script type="text/javascript">
            alert("Hello World");
        </script>
    </head>
    <body>
        <h1>This should alert "Hello World" when loading.</h1>
    </body>
</html>
```

Hello World – mit Funktion

- Definition als Funktion:

```
function hello(){
    return 'Hello World';
}
```

- syntaktisch kein Unterschied zwischen Prozeduren und Funktionen
- Kein Rückgabetyp
- Kommentare entweder einzeilig mit `// Kommentar`
  - Oder mehrzeilig mit `/*Kommentar*/`

assono

Variablen

- Werden durch einfache Zuweisung mit "=" ohne Typ angelegt

-> loose typing

- Empfohlen wird die Nutzung von "**var**"
  - Entspricht einer Neudefinition
  - Variablen-Scope nur Funktionskontext

Achtung: Javascript ist case-sensitive

Typen

- „loose typing“ heißt nicht untypisiert
- Es gibt folgende 6 Typen:
  - string
  - number
  - boolean
  - object (auch z.B. arrays)
  - `null` – Eigenschaften ohne Wert bzw. Funktionen ohne Rückgabewert
  - `undefined` – nicht definierte oder gelöschte Variable

typeof

`typeof` hilft nur bedingt

Typ	Ergebnis von `typeof`
object	'object'
function	'function'
array	'object'
number	'number'
string	'string'
boolean	'boolean'
`null`	'object'
`undefined`	'undefined'

Boolean

- Folgende Werte sind “falsch”
  - false
  - null
  - undefined
  - "" (leere Zeichenkette)
  - 0
  - `NaN`
- Alle anderen Werte sind “true”, auch
  - "0"
  - "false"

Number

- alle Zahlen sind 64-bit Gleitkommazahlen („double“)
- Rundungsdifferenzen durch Binär-Dezimal-Umwandlung
  - 4,98 + 0,2 = 5,180000000000001
- `NaN` = Not a Number
  - „ansteckend“
  - ungleich allem anderen (inklusive `NaN`)

  -> Test mit: `isNaN(Variablen-Name)`

Bedingte Ausführung - if...then...else

```
if(Bedingung){
    TuWas();
    UndNochWas();
}else if (Bedingung2)
    TuWasAnderes();
else
    DannHaltDas();
```

- Bedingung in Klammern
- Wenn zutreffend wird die nächste Anweisung ausgeführt
- Geschwungene Klammern fassen einen Block zu einer Anweisung zusammen

Switch

```
switch(Ausdruck){
    case wert1:
        TuDas();
        break;
    case wert2:
        TuDies();
        break;
    default:
        OderDas();
}
```

- Ohne die `break`-Anweisung wird der nächste `case` ausgeführt – sonst Beendigung der Switch-Anweisung
- Wenn nichts zutrifft, wird `default` ausgeführt.

assono

Vergleichsoperatoren

- `==` und `!=` vergleichen Werte
- `===` und `!==` vergleichen Werte unter Berücksichtigung des Typs
  - `1 == "1"` -> Wahr
  - `1=== "1"` -> Falsch
- <, <=, >, >=, ! usw. wie in LotusScript

Logische Operatoren

- `"Ausdruck1 && Ausdruck2"` gibt Ausdruck1 aus, wenn Ausdruck1 falsch ist, sonst Ausdruck2
- `"Ausdruck1 || Ausdruck2"` gibt Ausdruck1 aus, wenn wahr, sonst Ausdruck2

Schleifen - for

```
for([init];[Bedingung];[Increment]){
    Anweisungen
}
```

- `[init]` setze Initialwert der Inkrementierungsvariable
- `[Bedingung]` Austrittsbedingung
- `[Increment]` Verändere Inkrementierungsvariable
- Mit "`break`" unterbricht man auch ohne erfüllte Bedingung die Schleife
- Mit "`continue`" springt man zum nächsten Inkrement

Schleifen – while/do...while

```
while(Bedingung){
    Anweisung(en)
}
```

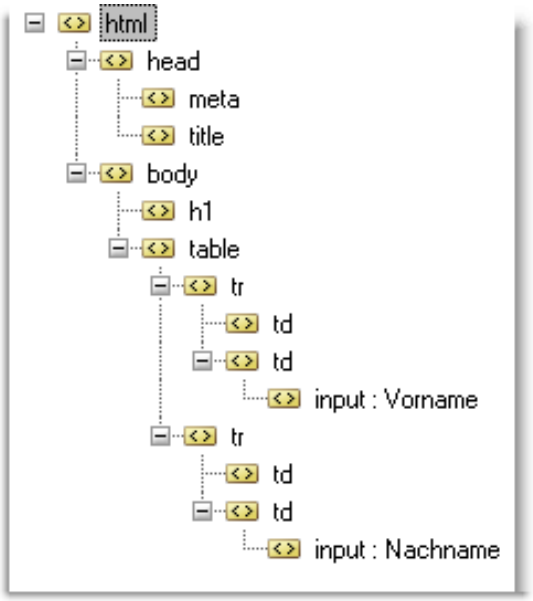
- Anweisung wird so lange ausgeführt bis die Bedingung nicht mehr erfüllt ist

```
do{
    Anweisung(en)
}while(Bedingung)
```

- Wie oben, aber die Anweisung wird mindestens einmal ausgeführt

DOM – Document Object Model

- Jedes HTML-Dokument lässt sich als Baum darstellen
- Die Wurzel heißt `"window"`
- Man kann auch mit `"document"` starten
- Jeder “Kindknoten” heißt `"element"`
- Elemente haben:
  - Attribute (z.B. `width`)
  - Namen (nicht eindeutig)
  - Id's (eindeutig)
  - Klassen (für CSS)
  - Events (z.B. `onclick`)
  - Werte



DOM – Elemente und Werte ermitteln

- `document.getElementById`: Element jeglichen Typs mit Hilfe der ID
- `document.getElementsByTagName`: Element eines bestimmten Typs
- `element.value` gibt den Wert zurück bzw. setzt ihn
- Elemente können auch direkt mit der ID angesprochen werden, wenn man den "Pfad" kennt
  - `document.Formname.ElementID`

assono

DOM - Manipulation

- `document.createElement(name)` erzeugt ein Element
- `element.appendChild(element)` hängt an das aktuelle Element ein Weiteres an
- `element.setAttribute(attributname, attributwert)` ändert das angegeben Attribut des Elementes
  - `element.getAttribute(attributName)` umgekehrt
- Mit `element.innerHTML` kann man neue Elemente (als string) einfügen

DOM – Style Attribut

Das Style-Attribute ist eine Besonderheit, da es mehrere Style-Informationen beinhalten kann

-> **`Style="width: 50px; font-size: bold;"`**

- `element.setAttribute("Style","width:60px")` wird alle anderen Styles überschreiben
- Mit `element.style.StyleName` erhält man direkten Zugriff auf den jeweiligen Style

-> **`elem.style.backgroundColor = "red";`** (oder #FF0000)

oder

-> **`elem.style.set/get/removeAttribute(AttributName[, Wert]);`**

-> **`elem.style[AttributName] = Wert;`**

Events

HTML-Elemente feuern events. Die Wichtigsten:

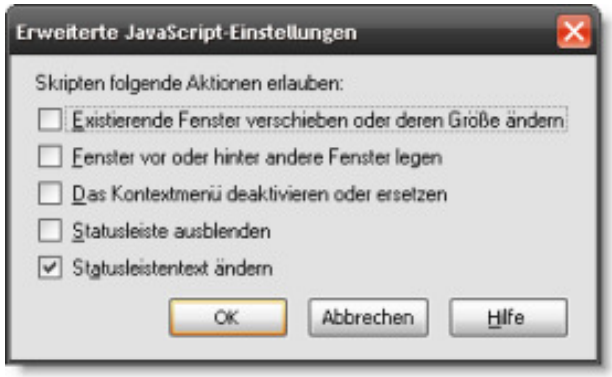
- Das Form-Element feuert `onsubmit()` ~ onQuerySave() ideal für client-seitige Validierung
- Buttons u.a. feuern `onclick()` -> zum Aufruf von JS-Funktionen
- Input-Felder feuern `onchange()` beim Verlassen des Feldes, wenn geändert -> gut für Input-Translation

Übung - Anmeldeformular

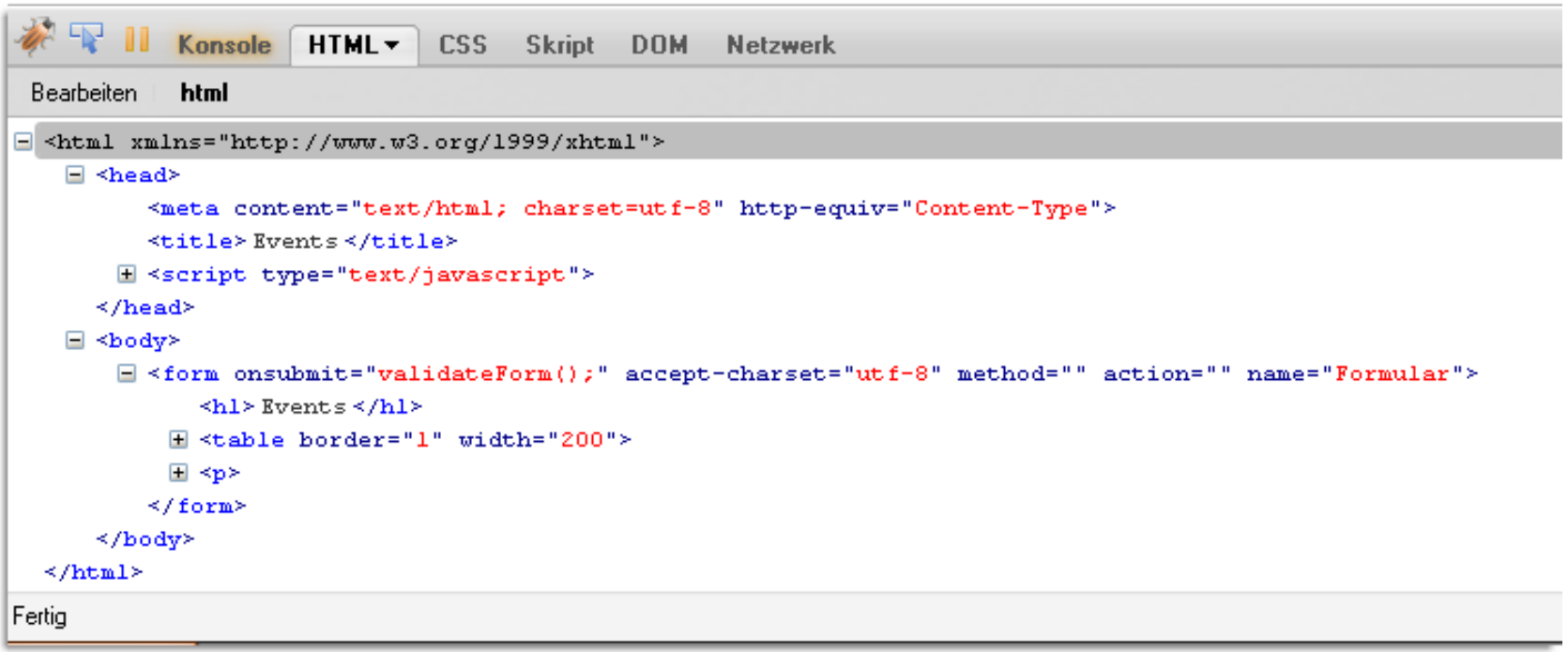
- Erstellt ein Formular mit 4 Felder:
  - Vorname
  - Nachname
  - Alter
  - Beruf
- Input-Translation für Alter (Ganze Zahl)
  - Nutze `parseInt(String)`, um die Zahl zu erzeugen
  - `Wert.ToFixed(Stellen)`, um die Zahl der Dezimalstellen festzulegen
- Der Vor- und Nachname muss eingetragen werden

Debugging

- `alert(String)` hilfreich, um Status von Variablen auszugeben, muss aber einzeln bestätigt werden
- `window.status` verändert die Statusbar, wenn erlaubt
- FireBug



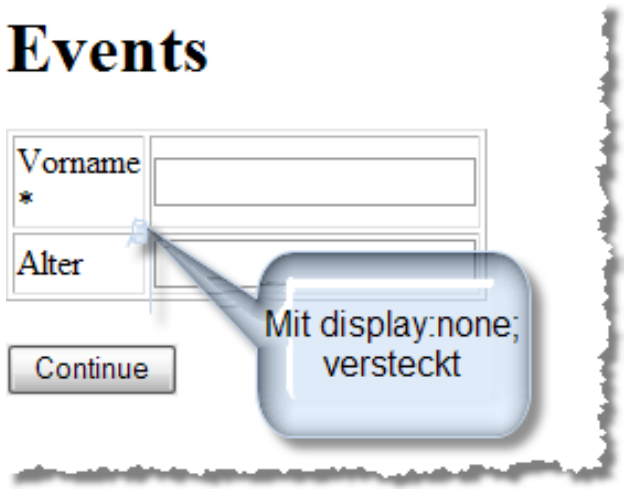
Demonstration Firebug



Hide-When – Visibility vs. Display

- Style visibility – verändert nur die Sichtbarkeit
  - hidden
  - visible
  - collapse wie hidden für Reihen und Spalten – nicht in IE
- `display:none` – verändert die "Flow"-Eigenschaft
  - Zum Anzeigen entweder Attribute löschen oder Anzeigeeigenschaft "inline", "block" o.ä. definieren

Visibility vs. Display



Übung mit Hide When

Wir erweitern die Maske von eben um ein Feld Zweiter Vorname, das versteckt ist bis der Vorname eingegeben wurde

- Visibility oder Display?

Script Bibliotheken

- Wiederverwertbar
- Austauschbar

```
<script type="text/javascript" src="Pfad/name.js"></script>
```

- Wenn die Datei “lokal” liegt, dann ist relative Referenzierung zu empfehlen
- Es können auch Webreferenzen angegeben werden
- In Notes befindet sich die Bibliothek im “Hauptverzeichnis”

Fehlerhandling: try..catch..finally

```
try{
    Anweisungen;
    throw new UserException();
}catch(e[if e == "ExceptionType"]){
    behandleFehler();
}finally{
    WirdAufJedenFallGetan();
}
```

- Fehler im `try` – block werden mit `catch` gefangen und können da behandelt werden (Log-Eintrag)
- Der Name ist frei-definierbar
- Im `finally`-block können Bedingungen für die weitere Bearbeitung gesetzt werden

assono

JS in Notes-Forms

- In Forms (Pages, Views etc.) gibt es JS-events entsprechend den events einer HTML-Seite
- Im JS Header werden JS-Funktionen oder Variablen definiert
- Im HTML Head können JS-Bibliotheken eingebunden werden
- In WebQuerySave kann man einen Agenten aufrufen, der für die server-seitige Validierung zuständig ist

Achtung: Javascript kann Browser-seitig abgestellt werden!

Exkurs: WebQuerySave

Server-seitige Validierung wird **dringend** empfohlen:

- WebQuerySave wird vor dem Speichern des Dokumentes gestartet
- Der Agent kann die Speicherung mit SaveOptions="0" verhindern
- Alle print-statements im Agenten werden an den Browser zurück gegeben –> HTML wird interpretiert
- Das Feld "$$return" auf der Form wird ausgewertet, wenn WebQuerySave keinen String-Output hat

Einsatz von Frameworks

- Frameworks bieten häufig Convenience-Funktionen
- Bügeln häufig Unterschiede zwischen Browsern aus (z.B. bei Ajax)
- Bieten zusätzliche Funktionen für UI-Manipulation wie z.B. D'n'D
- Immer darauf achten, dass das Framework aktuell ist und gepflegt wird
  - Dojo – wird auch in Notes verwendet
  - script.aculo.us
  - Prototype

Weitere Session zu JS

- Di. 9:00 Track 2: Dojo und Notes - Bernd Hort
- Di. 11:00 Track 2: JavaScript für Fortgeschrittene - Thomas Bahn
- Di. 14:00 Track 3: OpenNTF Projekte basierend auf XPages und Dojo - Niklas Heidloff
- Mi. 9:00 Track 4: Fehlerbehandlung in Formelsprache, LotusScript, Java und JavaScript - (Thomas Schulte) Bernd Hort

Ressourcen

- http://de.selfhtml.org/javascript/index.htm – nicht mehr ganz aktuell, aber gut strukturiert
- Dustin Diaz, Ross Harmes: Pro JavaScript Design Patterns
- “assono-Framework” auf OpenNTF.org für einige convenience-Funktionen und Modelvalidierung
- Und natürlich immer Leute fragen, die es wissen

Fragen?

jetzt stellen – oder später:

mnientit@assono.de
http://www.assono.de/blog
04307/900-406

Folien unter
http://www.assono.de/blog/