

Vorlesungsreihe
EwA

Javascript , Ajax und das Web 2.0

Prof. Dr.-Ing. Thomas Wiedemann
email: wiedem@informatik.htw-dresden.de



HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT DRESDEN (FH)
Fachbereich Informatik/Mathematik

Gliederung

- Javascript
 - Einführung / Historie, Abgrenzung zu Java
 - Grundlegende Technologien & Syntax
 - Beispiele
- Ajax
- Ausblick - Web 2.0

Quelle(n) :

[1] Selfhtml <http://de.selfhtml.org/>

Übersicht zu Javascript

- Javascript ist keine direkte Ableitung von Sun's JAVA sondern eine Eigenentwicklung von Netscape aus dem Jahre 1995
- Hauptziel ist die Client-seitige Manipulation des Dokumentes im Browser („Intelligenz“ und Dynamik im Browser)
- Im Rahmen des Browserkriegs zwischen Netscape und Microsoft wurde durch MS eine Konkurrenzsprache „Jscript“ eingeführt (zunehmend unbedeutender)
- Wie bei HTML bestehen zum Teil erhebliche Unterschiede bei der Unterstützung der Javascript-Versionen durch die Browser :
 - Die JavaScript-Version 1.3 ist in Netscape in den heute am meisten verbreiteten Netscape-Versionen 4.06 bis 4.7 implementiert (1998-2001). Der Internet Explorer versteht diese Sprachversion weitgehend seit Produktversion 5.0.
 - In die völlig neu programmierte Netscape-Version 6.0 wurde eine Version 1.5 von JavaScript integriert (2000-2001), die neben einigen wenigen JavaScript-eigenen Neuerungen vor allem das **Document Object Model (DOM) des W3C** umsetzt.
- **Fazit (nach [selfhtml]) : Es ist ratsam, benutzte Sprachbestandteile von JavaScript-Versionen jenseits der V1.2 besonders sorgfältig zu testen. Teilweise muß mehrfach für verschiedene Browser-Versionen programmiert werden.**

Einbindung von Javascript in HTML-Seiten

- Javascript-Code wird entweder zwischen `<script..>`-Tags (meist im Header oder zu Beginn) eingebunden oder aus einer externen Datei geladen (Erw. *.js) oder in HTML-Tags zum Aufruf komplexerer Funktionen verwendet

```
<html><head><title>Test</title>
<script type="text/javascript">
    alert("Hallo Welt!");
</script> </head> <body> </body>

<script src="quadrat.js"
type="text/javascript">
</script>

....
<input type="button" value="Quadrat
errechnen" onClick="Quadrat()" >
</html>
```

Datei: quadrat.js

```
function Quadrat()
{ var Ergebnis = document.Formular.Eingabe.value *
document.Formular.Eingabe.value; alert("Das Quadrat von " +
document.Formular.Eingabe.value + " = " + Ergebnis); }
```

Allgemeine Syntax-Regeln für JavaScript

- generelle Schreibweise an Java angelehnt, wenn auch nicht 100% ig
- Anweisungen werden mit ; abgeschlossen (nicht mehr zwingend notwendig, aber zwecks Kompatibilität zu Java sehr sinnvoll) `Quadrat = Zahl * Zahl;`
- Anweisungsblöcke zur Zusammenfassung größerer Anweisungsblöcke mit { } insbesondere bei Schleifen und Bedingungen
`if (Zahl > 1000) { Zahl = 0; Neustart(); }`
- Alle Schleifen und Bedingungen werden analog zu JAVA bzw. C unterstützt
`if () {} else {} / while () {}`
- Eigene Bezeichner analog zu JAVA: keine Leerzeichen / nur Buchstaben und Ziffern (Keine Umlaute!) und _ / erstes Zeichen muß ein Buchstabe sein / Groß- und Kleinschreibung werden unterschieden!
- Kommentare mit /* */ oder //
- Relativ lockere Typbindung von Variablen in JavaScript. Einfache Variablentypen, wie Zahlen, Zeichenketten oder Wahrheitswerte, werden lediglich nach numerischen und nicht-numerischen Variablen eingeteilt.
- Schlüsselwort var dient zur Definition von Variablen `var i = 0 ;`

Hilfreiche JavaScript-Befehle zum Testen

- Normale Messageboxen werden angezeigt mit
`<script type="text/javascript"> alert("Hallo Welt!" + zahl); </script>`
- Zur Interpretation von Rechenanweisungen zur Laufzeit kann mit eval ein mathematischer Ausdruck ausgewertet werden :
`function Rechne (Operation)
{ var Ergebnis = eval(Operation);
 alert("Ergebnis: " + Ergebnis); }`
- Fehler können mit onerror abgefangen werden :
`<script type="text/javascript">
 window.onerror = Fehlerbehandlung;
 function Fehlerbehandlung (Nachricht, Datei, Zeile) {
 Fehler = "Fehlermeldung:\n" + Nachricht + "\n" + Datei + "\n" + Zeile;
 alert(Fehler); return true; }
 nichtda(); // Aufruf einer nicht existenten Funktion
</script>`

Anbindung von Javascript an Browser-Events

- Das W3C-Konsortium hat in HTML entsprechende Event-Handler definiert, welche bei Auslösen der entsprechenden Aktion das Script aufrufen :
 - [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 bewegter 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)

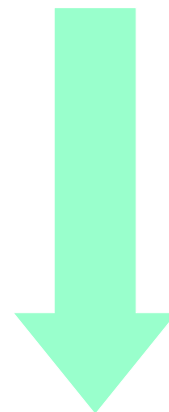
Entwicklung webbasierter Anwendungen

- Prof. T.Wiedemann

- HTW Dresden - Folie 7

Javascript - Objektreferenz

- Die Eventhandler stehen in Bezug zu allen sichtbaren und unsichtbaren Bestandteilen des Browsers. Diese Bestandteile werden durch die JavaScript-Objekthierarchie definiert:
 - [window](#) (Anzeigefenster), [frames](#) (Frame-Fenster)
 - [document](#) (Dokument im Anzeigefenster)**
 - [HTML-Elementobjekte](#) (Alle HTML-Elemente des Dokuments)
 - [node](#) (Alle Knoten des Elementenbaums)
 - [all](#) (Alle HTML-Elemente des Dokuments - Microsoft)
 - [style](#) (CSS-Attribute von HTML-Elementen)
 - [anchors](#) (Verweisanker im Dokument)
 - [applets](#) (Java-Applets im Dokument)
 - [forms](#) (Formulare im Dokument)
 - [elements](#) (Formularelemente eines Formulars)
 - [options](#) (Optionen einer Auswahlliste eines Formulars)
 - [images](#) (Grafikreferenzen im Dokument)
 - [layers](#) (Layer im Dokument - Netscape)
 - [links](#) (Verweise im Dokument)
 - [event](#) (Anwenderereignisse), [history](#) (besuchte Seiten)
 - [location](#) (URIs), [Array](#) (Ketten von Variablen)
 - [Boolean](#) (Ja/Nein-Variablen), [Date](#) (Datum und Uhrzeit)
 - [Function](#) (JavaScript-Funktionen)



Hierarchie

Entwicklung webbasierter Anwendungen

- Prof. T.Wiedemann



- HTW Dresden - Folie 8

Beispiele zur Arbeit mit Ereignissen und Objekten

- **Typische Aufgabe – Prüfung von Formulareingaben**

```
<script type="text/javascript"> function checkeFormular()
{ if (document.Formular.User.value == "")
{ alert("Bitte Namen eingeben!"); document.Formular.User.focus(); return false; }
if (document.Formular.Mail.value.indexOf("@") == -1)
{ alert("Keine E-Mail-Adresse!"); return false; } var chkZ = 1;
for (i = 0; i < document.Formular.Alter.value.length; ++i)
if (document.Formular.Alter.value.charAt(i) < "0" ||
    document.Formular.Alter.value.charAt(i) > "9") chkZ = -1;
if (chkZ == -1) { alert("Altersangabe keine Zahl!");
    document.Formular.Alter.focus(); return false; } } </script> </head>

<body> <form name="F1" action=http://.../f.php
    method="post" onsubmit="return checkeFormular()">
Name: <input type="text" size="40" name="User">
E-Mail: <input type="text" size="40" name="Mail">
Alter: <input type="text" size="40" name="Alter">
Formular: <input type="submit" value="Absenden">
<input type="reset" value="Abbrechen"> </pre> </form>
```



Beispiele und Details

- Alle weiteren Details und Beispiele zu Javascript
 - Grundlagen :
<http://de.selfhtml.org/javascript/index.htm>
 - Größere Anwendungsbeispiele :
<http://de.selfhtml.org/javascript/beispiele/index.htm>

Ajax (Asynchronous JavaScripting and XML)

- kombiniert JavaScript, HTML, DHTML, DOM und XML
- erzeugt stark interaktive Seiten und ermöglicht damit flüssigeres Arbeiten
- keine manuelle Interaktion des Users mit dem Server, sondern eigenständige Kommunikation von Javascript mit dem Server
- Ajax-Kommunikation durch das XMLHttpRequest-Objekt mit einem meist asynchronen XML-Datenaustausch
- **XMLHttpRequest-Objekt** ist verfügbar ab Microsoft Internet Explorer 5.0, Mozilla Firefox 1.0, Netscape 7.1, Apple Safari 1.2, Opera Mobile Browser 8.0
- **Achtung : leider noch unterschiedliche Objektreferenzen auf das XMLHttpRequest-Objekt !!**

Methoden von XMLHttpRequest

Die wichtigsten Methoden von XMLHttpRequest :

- Öffnen einer Verbindung zum Server
open(httpReqMeth, url, async)
open(httpReqMeth, url, async, usr, pwd)
httpReqMeth - definiert http-Methode ('GET', 'POST', 'PUT')
url = URL des Dienstes
async = true für asynchrone Kommunikation (Client wartet nicht auf Antwort, sondern es wird Callback-Funktion später aktiviert)
- Absenden eines Request
send(postReq) ; - postReq = null für 'GET'-Anfragen oder Key-Value-Paare für 'POST'-Anfragen (z.B. "Key1=Value1&Key2=2")
- **abort()** - Bricht eine aktuell laufende Anfrage ab
- **setRequestHeader(key, value)** - fügt dem HTTP-Header Werte zu

Die wichtigsten Attribute von XMLHttpRequest

- **onreadystatechange** - verweist auf die Event-Handler-Callback-Methode, welche bei Zustandsänderungen des XMLHttpRequest-Objekts aufgerufen wird (siehe ,
- **readyState** – aktueller Status des Request mit folgenden Werten :
0 = uninitialized 1 = loading 2 = loaded
3 = interactive **4 = complete**
- **status** - das Ergebnis des http-Request (= http-Status)
 - z.B.: **200 = OK** 404 = Not Found
- **statusText** - der HTTP-Status als Textmitteilung
- **responseText** - die Serverantwort als einfacher Text
- **responseXML** - die Serverantwort im XML-Format

Die Grundstruktur eines Ajax-Request

```
<script language="JavaScript" type="text/javascript">
var url = "http://localhost/checkiptxt.jsp"; var req;
function starteAjax()
{ try { if( window.XMLHttpRequest )
    { req = new XMLHttpRequest(); } // Version für Firefox & Co.
    else if( window.ActiveXObject )
        { req = new ActiveXObject( "Microsoft.XMLHTTP" ); } // IE
    else { alert( "Ihr Webbrowser unterstützt leider kein Ajax!" ); }
    req.open( "GET", url, true );
    req.onreadystatechange = meineCallbackFkt;
    req.send( null );
    } catch( e ) { alert( "Fehler: " + e ); } }
function meineCallbackFkt()
{ if( 4 == req.readyState ) { if( 200 != req.status )
    { alert( "Fehler " + req.status + ": " + req.statusText ); }
    else { alert( req.responseText ); } } }
</script>
```

Ajax-Request mit XML-Antwort

- Request analog zu vorheriger Seite und Anforderung XML-Dok.

```
if( req.overrideMimeType ) { req.overrideMimeType( 'text/xml' ); }
```

- Neue Auswertung in der Callback-Funktion :

```
function meineCallbackFkt()
```

```
{ if( 4 == req.readyState ) { if( 200 != req.status )
```

```
    { alert( "Fehler " + req.status + ": " + req.statusText ); }
```

```
    else { ergebnis = req.responseXML.documentElement;
```

```
    // Hole Werte aus dem XML-Response
```

```
    var zahlAusgabe = ergebnis.getElementsByTagName('zahl')[0].firstChild.data;
```

```
    var textAusgabe = ergebnis.getElementsByTagName('text')[0].firstChild.data;
```

```
    var ipAusgabe = ergebnis.getElementsByTagName('ip')[0].firstChild.data;
```

```
    // Schreibe Werte in die HTML-Seite
```

```
    document.getElementById("zahlAusgabe").value = zahlAusgabe;
```

```
    document.getElementById("textAusgabe").value = textAusgabe;
```

```
    document.getElementById("ipAusgabe").innerHTML = ipAusgabe;
```

```
}}}
```

Demo Ajax-Request für Auto-Vervollständigen

1. HTMLSeite mit Formular :

```
<form name="formular" action="...">
```

```
  <input type="text" id="eingabefeld" onKeyUp="meinAjaxAufruf( this.value )"/> <br>
```

```
  <div id="auswahlbox" #223377;"></div> </form>
```

2. Ajax - Request mit aktuellem Inhalt des eingabe-Feldes

```
document.formular.eingabefeld.focus();
```

```
var url = "autovervollstaendigung.jsp?eingabe=" + escape( eingabetext ); req.open( "GET", url, true );
```

3. Ajax – Response in Formular einbauen

```
var text = req.responseText;
```

```
if( text != "" )
```

```
{ auswahlarray = text.split( ";" );
```

```
  for( var idx in auswahlarray )
```

```
    { auswahlinhalt += "<a href='javascript:meinMausklick(" + idx + ")' id='" + idx;
```

```
    auswahlinhalt += "' class='aw' onmouseover=sel(\""+idx+"\")>";
```

```
    auswahlinhalt += auswahlarray[idx] + "</a>"; }
```

```
document.getElementById( "auswahlbox" ).innerHTML = auswahlinhalt; }
```


Generelle Ajax- Kritik

Vorteile

- im Gegensatz zu Flash oder ähnlichen Technologien wird kein Browser-Plugin benötigt, auch unabhängig von Betriebssystem
- Schnelleres, flüssigeres Arbeiten (kein unnötiges Laden von statischen Daten bei erneuten Request -> Einsparung von Bandbreite)

Nachteile :

- **JavaScript-Unterstützung muss aktiviert sein !**
- Noch Probleme mit unterschiedlichen Browserversionen (sollte sich durch W3C-Standardisierung legen)

Weitere Ajax- Kritik (ggf. technisch lösbar)

Generell umfangreiche Tests erforderlich

- ggf. Erleichterung durch entsprechende Frameworks (Fehlerhandlg.)
- siehe Dojo-Toolkit - <http://dojotoolkit.org/projects/core>

Probleme mit Browserversionen

- Durch Fallunterscheidungen auf Clientseite lösbar (ineffizient)
- -> Server-seitige Browsererkennung und spezifische Javascripte.

Verwendung der „Zurück“-Schaltfläche

- Funktionalität der „Zurück“- **Schaltfläche kann nicht** gewährleistet werden, da diese nicht über Ajax-Aktionen informiert wird
- Lösung durch Füllung von zusätzlichen Inline-Frame-Elementen oder speziellen Rückruf-Funktionen (bei Frameworks)

Bandbreitenprobleme bei ständigem Polling

- Da nur Kommunikation von Client-> Server->Server muss Client bei Verdacht auf Serveraktual. ggf.- pollen -> Netzlast !!!!
- Lösungen: Serverresponse künstlich verzögern bis zum Eintreten des Ereignisses oder eines Timeout's

Weitere Ajax- Kritik (ggf. technisch lösbar)

Analog zu den vorherigen Problemen :

- Lesezeichenspeicherung ?
- Barrierefreiheit ?
- Suchmaschinen erkennen die dyn. Ajax-Inhalte nicht

Lösungen :

- zusätzliche Metatags oder Überschriften-Elemente für die Indizierung
 - Zusätzliche (unsichtbare) Links werden auf der Webseite für die Suchroboter einer Suchmaschine gedacht sind.
 - zweite Webseite mit statischen Links wird entworfen. Diese ist für eine Suchmaschine voll zugänglich (Achtung: als Cloaking einstufbar)
 - Neu: Ajax-Interpreter in den Suchmaschinen selbst (???)
- **Gesamtfazit:**
 - Ajax ist eine interessante Technologie für Anwendungen mit hohem Interaktionsgrad und sollte in der Entwicklung beobachtet werden.

Ajax- Demos

- Im Netz verfügbare Demos und Details

http://www.ajax-net.de/index.php?option=com_wrapper&Itemid=62

http://de.wikipedia.org/wiki/Ajax_%28Programmierung%29

Web 2.0 – Übergang zu neuen Anwendungstypen

Entsprechend der Grundprinzipien erfolgt ein Übergang :

<u>web 1.0</u>	→	<u>web 2.0</u>
DoubleClick	→	Google AdSense
Ofoto (jetzt Kodakgallery)	→	Flickr
mp3.com*	→	Napster
Britannica Online	→	Wikipedia
personal websites	→	blogging (Persönliche Logs)
Evite →		Upcoming.org and EVDB
domain name speculation	→	search engine optimization
page views	→	cost per click (neues Abrechnungsmodell)
screen scraping	→	web services (Web-Dienste)
publishing	→	participation
content management systems	→	wikis
directories ("taxonomy")	→	tagging ("folksonomy" = social classification)
stickiness	→	syndication (Mehrfachverwendung von Inhalten)

Entwicklung webbasierter Anwendungen - Prof. T.Wiedemann - HTW Dresden - Folie 23

Web 2.0 – die interessantesten Technologien

SaaS – Software as a Service (früher ASP : Application Service Provider)

Basisidee Software auf Mietbasis

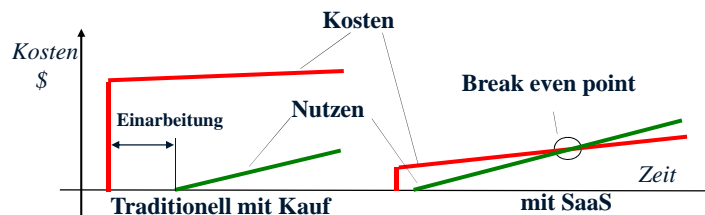
- alle anfallenden Kosten werden auf eine monatliche Rate umgerechnet, ggf. auch Pay per use (z.B. Stundenweise Bezahlung oder Kostensatz pro FEM-Berechnung)
- Dienstleister übernimmt auch Wartung und Risiko von Hardware-Ausfällen

Vorteile :

- für Gelegenheitsnutzer sehr viel günstiger als Kauf einer Dauerlizenz
- Kurzfristig auch größere Anzahl von Lizenzen nutzbar (Stoßgeschäft)

Nachteile:

- Netz muß immer vorhanden und performant sein
- Hohes Vertrauen in Dienstleister nötig (Pleite ?, Datenschutz ?)

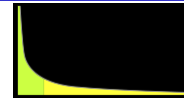


Entwicklung webbasierter Anwendungen - Prof. T.Wiedemann - HTW Dresden - Folie 24

Web 2.0 – Long tail und Virales Marketing

Long Tail : englisch für „langer Schwanz“)

- Begriff geprägt von Chris Anderson (Chefred. Wired Magazine)
- Konzept: Nutzung des Internet und Web 2.0 für den Vertrieb (oder auch Beschäftigung allg.) mit von Nischenprodukten oder exotischen Themen
- Durch Internetshops kann auch mit selten verkauften Produkten in der Gesamtsumme mehr Umsatz gemacht werden als mit wenigen Bestsellern
- Basis: Lösung von geografischen und traditionellen Restriktionen (Weltvertrieb mit UPS aus einer Scheune in den Bergen)
- Auch als politisches Konzept : Demokratisierung der Produktionsmittel (Softwareentw. durch Jedermann, Wikipedia : Lexikon für alle durch alle) Demokratisierung des Vertriebes [Bearbeiten]



Virales Marketing :

- neue Werbung mittels „Mund-zu-Mund-Propaganda“ im Web (gewisse Ähnlichkeit mit alten Kettenbriefsystemen ohne Bezahlzwang)
- auch für politische Themen und Kampagnen
- setzt einen guten Mix aus gutem Inhalt und cleverer Unterhaltung

Entwicklung webbasierter Anwendungen - Prof. T.Wiedemann - HTW Dresden - Folie 25

Zusammenfassung

- Web 2.0 als Sammelbecken neue Ansätze mit kommerziell erfolgreichen Systemen

Eine sichere Prognose erscheint fast unmöglich :

- **Optimistisch**

- Zukünftig noch mehr stark interaktive Anwendungen im Web
- bisherige Desktopanwendungen (Zukunft von windows? Und anderen OS) werden mit Browseranwendungen verschmelzen
- mobile Geräte werden dominieren (Unabhängig von konkreter HW !)
- Web 3.0 überall und für jeden ??

- **Pessimistisch :**

- große Firmen versuchen (wieder) durch eigene Teilstandards ihre Marktposition zu verbessern (-> starke Inkompatibilitäten und viele Probleme im Detail)

Entwicklung webbasierter Anwendungen - Prof. T.Wiedemann - HTW Dresden - Folie 26