WebTech (3)

Prof. Dr.-Ing. S. Gössner

University of Applied Sciences Lippe & Höxter

Inhalt

Web-Programmiersprachen

HTML und XML werden als *deklarative Sprachen bezeichnet. Es handelt sich hierbei nicht um sog. Programmiersprachen. Diese sind vielmehr* Befehlssprachen *oder* imperative Sprachen_.

- Imperative Programmiersprachen können auf dem Webserver zur Erstellung einer Webseite auf Anforderung eingesetzt werden.
- Programmiersprachen können andererseits zur Steuerung einer Programmlogik innerhalb einer Webseite verwendet werden.
- Wir unterscheiden dementsprechend zwischen serverseitigen und clientseitigen Programmiersprachen.

Serverseitige Programmiersprachen

- Auf der Seite des Webservers kann grundsätzlich jede Programmiersprache eingesetzt werden, wenn sie nur das zugrundeliegende Betriebssystem unterstützt.
- In der Praxis haben sich allerdings einige durchgesetzt:
  - Perl
  - PHP
  - Java
  - Python
  - Basic
  - C/C++
- Es überwiegen hier die Interpretersprachen
- Aufgabe der hier erstellten Programme ist
  - der Zusammenbau der angeforderten Webseite kurz vor der Auslieferung an den Browser.
  - Zugriff auf gewünschte Datenbankeinträge (SQL).
  - Daten von anderen Webservern anfordern.

Clientseitige Programmiersprachen

- Diese Programmiersprachen müssen innerhalb des Browsers verfügbar sein.
- *Javascript* ist standardisiert (*ECMA 262*) und dominierend.
- *VBScript* ist proprietär und praktisch bedeutungslos.
- *Java Applets* sind ebenfalls proprietär und heute bedeutungslos.
- Programme innerhalb des Browsers dürfen auf keinen Fall auf das Datei- und Betriebssystem des Benutzers zugreifen (Ausnahme: *Cookies*).
- Vor einigen Jahren galt: *Javascript + HTML = DHTML*
- *Javascript* hatte lange Zeit ein negatives Image:
  - Sprache für Spielereien.
  - stellt Sicherheitslücke dar; diese Generalisierung basiert jedoch auf Fehlern (*Bugs*) in der Implementierung von Browserkomponenten oder der Sprache selbst.
  - Fehlende professionelle Literatur und vorbildliche Beispielanwendungen.
- Gegenwärtig gewinnt *Javascript* zunehmend an Ansehen und ist für moderne Webanwendungen unverzichtbar (*Ajax*).

Javascript - Eigenschaften

Javascript …

- ist eine leistungsstarke Programmiersprache bzw. …
- eine Interpretersprache und benötigt eine Laufzeitumgebung *(Browser)*
- besitzt keine eigenen Ein-/Ausgabemöglichkeiten *(Sandbox)* und kann lediglich die der jeweiligen Umgebung nutzen.
- ist objektbasiert.
- besitzt Eigenschaften einer *funktionalen* Programmiersprache *(Closures)*.
- ist mittlerweile standardisiert - *ECMA-262*

Programmstruktur

- Programm
  - Block
    - Block
    - Funktion
    - Anweisung
    - Kommentar
  - Funktion
    - Block
    - Funktion
    - Anweisung
    - Kommentar
  - Anweisung
    - Literal
    - Variable
    - Schlüsselwort
    - Operator
  - Kommentar
    - einzeilig
    - mehrzeilig

Datentypen

Javascript ist eine schwach typisierte Sprache *(weakly typed)*. Es gibt folgende sieben Datentypen:

- Number
- String
- Boolean
- undefined
- null
- Object
- Function

Literale

Literale sind konstante Ausdrücke eines bestimmten Typs.

- ganzzahlige Literale *(Number)*
  - Dezimalzahl
    - Syntax: `/[+-]?[0-9]+/`
    - Bsp.: *(42, +2006, -467)*
  - Hexadezimalzahl
    - Syntax: `/0[xX][0-9a-fA-F]+/`
    - Bsp.: *(0x42, 0X5AE, 0x2cfd)*
- Gleitkommazahl *(Number)*
  - Syntax: `/[+-]?[0-9]*\.?[0-9]*([eE][+-]?[0-9]+)?/`
  - Bsp.: *(3.14, -.1, +12., 0.462e-3)*
- Zeichenkette *(String)*
  - Syntax: `/(".*")|('.*')/`
  - Bsp.: *("Hallo", "", 'wie geht\'s')*
- Wahrheitswert *(Boolean)*
  - Syntax: `/(true\|false)/`
- Object:
  - Syntax: `/{(name:expr,)*(name:expr)?}/`
  - Bsp.: *({d:"mon",h:11,m:59,s:30})*
- Array:
  - Syntax: `/\[(expr,)*expr?\]/`
  - Bsp.: *([8,-25,"pi",false])*

Variablen

Variablen *(Bezeichner)* sind Symbole für den Inhalt von Speicherbereichen in einem Programm. Diese Speicherbereiche werden gemäss des zugewiesenen Datentyps interpretiert.

Variablen ...

- müssen **deklariert** *(vereinbart)* werden.
- können **definiert** werden *(Wertzuweisung)*.
- -namen müssen der Syntax `/[a-zA-Z_$][a-zA-Z0-9_$]*/` gehorchen. (Ab Javascript 1.5 sind auch *Unicode*-Zeichen zugelassen).
- können ihren Datentyp während ihrer Lebensdauer ändern *(schwach typisiert)*.
- erhalten bei einer reinen Deklaration automatisch den Wert `undefined`.
- sollten mit dem *Schlüsselwort* `var` deklariert oder definiert werden.

Beispiel:

```
var pi=3.14, s="Hallo", u;
var arr = [24, 35, -48],
    obj = {mat_nr: 1996746930,
           name: "Müller, Elfriede"};
var pi2 = 2*pi, piroot = Math.sqrt(pi);
```

Operatoren

Operatoren führen eine Operation mit einem oder mehreren Operanden durch.

- unärer Operator: `@X`
- binärer Operator: `X @ Y`
- ternärer Operator `X ? Y : Z`

Operator type	Operators
comma	,
assignment	= += -= *= /= %= <<= >>= >>>= &= ^= |=
conditional	?:
logical-or	||
logical-and	&&
bitwise-or	|
bitwise-xor	^
bitwise-and	&
equality	== != === !==
relational	< <= > >= in instanceof
bitwise shift	<< >> >>>
addition/subtraction	+ -
multiply/divide	* / %

negation/increment	! ~ - + ++ - - typeof void delete
call / create instance	() new
member	. []

[Quelle: MozillaDeveloper Center]

Ausdrücke

Ein *Ausdruck (expression)* …

- ist Teil einer *Anweisung*.
- ist eine gültige Sequenz von Literalen, Variablen, Operatoren und/oder Ausdrücken.
- resultiert in einem Wert mit definierten Datentyp *number, string, boolean* oder *object*.
  `( 8*5 + 2 )`
- kann einer Variablen einen Wert zuweisen.
  `( x = 98%50 - 6 )`
- heisst je nach Datentyp des Wertes
  - arithmetischer Ausdruck
  - Stringausdruck
  - logischer Ausdruck
  - Objektausdruck

Anweisungen

Eine Anweisung *(statement)* …

- ist die notwendigerweise minimale Einheit zur Programmausführung.
- hat selbst keinen Wert bzw. Datentyp.
- zur Programmverzweigung ist eine *Bedingungsanweisung*.
- für eine Programmschleife ist eine *Wiederholungsanweisung*.

```
var s = "hallo", vocals=0;

for (var i=0; i<s.length; i++) {
   switch (s.charAt(i)) {
      case "a": case"A":
      case "e": case"E":
      case "i": case"I":
      case "o": case"O":
      case "u": case"U": vocals++; break;
      default: break;
   }
}

window.alert(vocals + " Vokale gefunden!");
```

Bedingungsanweisungen

Eine Bedingungsanweisung *(conditional expression)* …

- bewirkt eine Verzweigung im Programmablauf in Abhängigkeit von einem Booleschen Ausdruck.
- kann in zweierlei Form auftreten.

if … else

```
if (bedingung) {
   anweisung*
}
else {
   anweisung*
}
```

switch

```
switch(ausdruck) {
   case literal1:
      anweisung*
      break;
   ...
   case literalN:
      anweisung*
      break;
   default:
      anweisung*
      break;
}
```


Wiederholungsanweisungen

Eine Wiederholungsanweisung *(loop)* …

- erlaubt die mehrfach, periodische Abarbeitung von Programmteilen.
- kann in vier unterschiedlichen Schreibweisen verwendet werden.

do … while

```
do {
   anweisung*
}
while (bedingung);
```

for

```
for (initialisierung; bedingung; inkrementierung) {
   anweisung*
}
```

for … in

```
for (eigenschaft in object) {
   anweisung*
}
```

while

```
while (bedingung) {
   anweisung*
}
```


Kommentare

Ein Kommentar *(comment)* ...

- wird vom Interpreter nicht verarbeitet (überlesen).
- dient dem aktuellen oder einem nachfolgenden Programmierer zum besseren Verständnis von Programmteilen.
- wird gerne benutzt, um temporär einen Programmteil auszublenden.
- kann einzeilig oder mehrzeilig kodiert werden.
- in mehrzeiliger Form kann er nicht mit einem weiteren mehrzeiligen geschachtelt werden.

```
// Dies ist ein *einzeiliger* Kommentar
var x = 42;
/* Und nun
   ein mehr-
   zeiliger
   Kommentar */
```

Funktionen

Eine Funktion *(function)* …

- ist der Baustein zur strukturierten Programmierung.
- unterstützt die Wiederverwendung von Programmcode.
- wird genau **einmal** unter einem eindeutigen Namen definiert.
- kann dann beliebig oft aufgerufen werden.
- kann am Ende seiner Abarbeitung einen Wert mittels `return` zurückliefern.
- kann namentlich einer Variablen zugewiesen werden.
- hat die Syntax
  `function name(argument*) { anweisung* }`

Beispiel:

```
function TagPreis(artikel) { return 4.90; }
function NachtPreis(artikel) { return 7.90; }

var hour = 19,
    Preis = hour > 20 ? TagPreis
                      : NachtPreis;
Preis("bier");
```


Objekte

- Javascript selbst stellt nur einige wenige Objekte zur Verfügung
  (Array, Boolean, Date, Function, Global, Math, Number, RegExp, String)
- Die jeweilige Laufzeitumgebung bietet im Allgemeinen jedoch eine Schnittstelle *(API)* zu zahlreichen problemspezifischen Objekten *(z.B. Browser)*. Objekte besitzen Eigenschaften *(properties)* und Methoden *(methods)*.
- Objekte können über Literale definiert werden.
- bereits existierende Objekte können um Eigenschaften und Methoden erweitert werden.

Beispiel:

```
// static circle object
var circle = {
  x:25, y:35,
  r:10,
  color:"green",
  length: function() { return 2*Math.PI*circle.r; }
  area: function() { return Math.PI*circle.r*circle.r; }
};
window.alert(circle.area());

// circle constructor
function Circle(x, y, r, color) {
  this.x = x; this.y = y;
  this.r = r;
  this.color = color;
}
Circle.prototype = {
  length: function() {return 2*Math.PI*this.r; }
  area: function() {return Math.PI*this.r*this.r; }
};

var c1 = new Circle(20,30,5,"red"),
    c2 = new Circle(120,30,7,"blue");
```


Einbindung von Javascript in HTML

Es gibt genau drei Möglichkeiten, Javascript Programmcode in HTML und XML einzubinden.

- Referenzierung auf eine oder mehrere externe Datei(en) *(*.js)*.
- Inline Kodierung des Programmtextes an einer oder mehreren Stellen.
- Anweisung(en) als Wert von Elementattributen.

Beispiel:

```
<html>
  <head>
    <title>Javascript Einbindung</title>
    <script type="text/javascript" src="extern.js"> </script>
    <script type="text/javascript">
      var a = 3, z = 14*a;
      function init() { /* ... */ }
      ...
    </script>
  </head>
  <body onload="init();">
     <!-- ... -->
  </body>
</html>
```

Aufruf von Javascript-Anweisungen in HTML

Eine Javascript Anweisung wird

- während des Ladevorgangs
  - direkt ausgeführt, wenn …
  - sie nicht innerhalb einer Funktion kodiert ist.
- als Antwort auf eine Benutzerinteraktion über einen Event-Handler ausgeführt.

Beispiel: Beispiel:

```
<html>
  <head>
    <title>Javascript Einbindung</title>
    <script type="text/javascript" src="extern.js"> </script>
    <script type="text/javascript">
      var c = 2 + 2 + "2";
    </script>
  </head>
  <body onclick="alert("84"*1/2);">
    <script type="text/javascript">
      document.write(23%8*6);
    </script>
     <!-- ... -->
  </body>
</html>
```


HTML Event-Handler

- onblur
- onclick
- ondblclick
- onfocus
- onkeydown
- onkeypress
- onkeyup
- onload
- onmousedown
- onmousemove
- onmouseout
- onmouseover
- onmouseup
- onresize
- onunload