INP (3)

Prof. Dr.-Ing. S. Gössner

University of Applied Sciences Lippe & Höxter

Inhalt

Javascript

> ***Javascript*** is the world's most misunderstood programming language.
>
> Douglas Crockford

Javascript - Historie

Javascript …

- wurde 1995 von Brendan Eich , einem Ingenieur von Netscape erfunden.
- wurde wegen der damaligen Popularität *Java*'s von *Livescript* in *Javascript* umbenannt.
- wurde 1996 von *Microsoft* in einer eigenen Variante dieser Webprogrammiersprache als *Jscript* herausgebracht.
- ist 1997 von *Netscape* an das Standardisierungsgremium ECMA übergeben worden und ist heute eine ISO-Norm *(ISO/IEC 16262)*.
- 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).
- hat die gegenwärtige Versionsnummer 1.5 *(Ecmascript Edition 3)*.
- ist annähernd auf jedem Computer der Welt installiert.
- wird innerhalb unterschiedlicher Softwarepakete benutzt
  - Windows Scripting Host
  - Adobe/Macromedia Flash
  - Adobe Acrobat
  - Photoshop
  - Browser

Javascript Eigenschaften

Javascript …

- ist eine leistungsstarke, clientseitige Webprogrammiersprache
- ist eine *Interpretersprache* und benötigt eine Laufzeitumgebung *(Browser)*.
- ist eine *prozedurale und objektorientierte* Programmiersprache.
- besitzt eine C-ähnliche Syntax.
- weist Eigenschaften einer funktionalen Programmiersprache auf (Closures).

Einbindung in HTML

Javascript-Programmcode wird mittels des `script` Elements eingebunden.

```
<script type="text/javascript"> </script>
```

Javascript wird …

- üblicherweise in den *head*-Bereich des *HTML*-Dokuments eingebunden.
- in eher seltenen Fällen direkt in den *body*-Breich geschrieben.
- gelegentlich als Quelltext innerhalb von Attributwerten verwendet *(Event Handler)*.
- als *Inline*-Quellcode direkt innerhalb von *HTML* notiert, oder …
- über eine externen Datei referenziert.

Interne Javascript Einbindung

```
<html>
  <head>
    <title>Webseite mit Javascript</title>
    <script type="text/javascript">
      window.alert("Hello World");
    </script>
  </head>
  <body>
    ...
  </body>
</html>
```

Externe Javascript Referenz

Datei: Script.js

```
window.alert("Hello World");
```

Datei: Page.html

```
<html>
  <head>
    <title>Webseite mit Javascript</title>
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body>
    ...
  </body>
</html>
```

Eingabe und Ausgabe

Im Gegensatz zu anderen Programmiersprachen besitzt Javascript keine eigene Möglichkeit der Datenein- und -ausgabe. Hierzu werden die Fähigkeiten der jeweiligen Programmierumgebung genutzt.

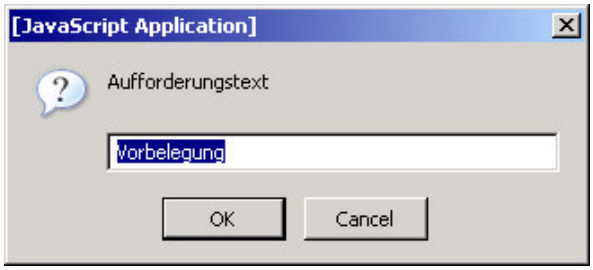
Der Browser bietet:

- Eingabe
  - Methode `window.prompt`
  - *HTML*-Formularelemente
- Ausgabe
  - Methode `window.write` *(direkt in Webseite, nur während des Seitenaufbaus)*.
  - Methode `window.alert` *(als Meldungsfenster)*.
  - mittels des DOM *(direkt in Webseite, auch nach erfolgten Seitenaufbau)*.

Eingabe

Eingabe mittels `windows.prompt`

```
var str = window.prompt("Aufforderungstext", "Vorbelegung");
```



Ausgabe

Ausgabe mittels `document.write`

```
<html><head><body><pre><script type="text/javascript">

document.write("Hello world");

</script></pre></body></html>
```

Ausgabe mittels `window.alert`

```
<html><head><body><pre><script type="text/javascript">

window.alert("Hello world");

</script></pre></body></html>
```

Programmstruktur

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

Programmablauf

Der Ablauf eines Javascript Programms beginnt mit der ersten Zeile und endet mit der Letzten, es sei denn, der Ablauf wird durch Verzweigungen, Schleifen und/oder Funktionen anders organisiert.

```
var a;
a = 1;
a = 2;
a = 3;
a = 4;
a = 5;
a = 6;
a = 7;
a = 8;
// ...
a = 999;
```

Datentypen

Javascript ist eine ***schwach typisierte*** Sprache, im Gegensatz zu den meisten kompilierenden Sprachen, welche ***streng typisiert*** sind.

Dies bedeutet, dass Variablen ihren Datentyp während der Laufzeit ändern können.

Javascript besitzt die folgenden Datentypen:

- Number
- String
- Boolean
- undefined
- null
- Function
- Object
    - Array
    - Date
    - RegExp

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);
```

Datentyp Number

Es wird nicht wie in anderen Sprachen zwischen ganzzahligen *(integer)* und Gleitkommawerten *(float)* unterschieden.

Alle Zahlen sind

> double-precision 64-bit format IEEE 754 values

Es gibt jedoch *integer* und *float* Literale.

Literale	Syntax	Beispiele
dezimal integer	`/[+-]?[0-9]+/`	`42`, `+2006`, `-744`
hexadezimal integer	`/0[xX][0-9a-fA-F]+/`	`0x42`, `0X5AE`, `0x2cfd`
float	`/[+-]?[0-9]*\.?[0-9]*([eE][+-]?[0-9]+)?/`	`3.14`, `-.1`, `+12.`, `0.462e-3`, `1E2`

Javascript kennt zwei weitere Symbole:

- `NaN` … *Not a Number* … resultiert aus: `parseInt("Hello")`
- `Infinity` … resultiert aus: `1 / 0`

Arithmetische Operatoren

Javascript besitzt die arithmetischen Operatoren der C-Sprachfamilie.

Operator	Symbol	Beispiel
Addition	+	`39 + 3 // 42`
Subtraktion	-	`75 - 13 // 42`
Multiplikation	*	`6 * 7 // 42`
Division	/	`126 / 3 // 42`
Modulo	%	`170 % 64 // 42`
Increment	++	`var x=41; ++x // 42`
Decrement	--	`var x=43; --x // 42`

Vorrangregeln

Es gilt die vertraute Regel der Mathematik: *"Punkt- vor Strichrechnung"*, wobei der *Modulo*–Operator zu den *Punkt*–Operatoren gehört.

Inkrement– und *Dekrement*–Operator besitzen Vorrang hinsichtl. der anderen Operatoren.

Um die Vorangregeln zu umgehen, werden *(runde)* Klammern – wie in der Mathematik – verwendet.

Beachte:

Javascript verwendet keine interne Rundungsarithmetik für Gleitkommazahlen, d.h. durch Rundungsfehler der Binärarithmetik sind Ergebnisse wie

```
0.1 + 0.2 = 0.30000000000000004
```

möglich.

Arithmetische Konstante und Funktionen

Aufruf	(Rückgabe)Wert
`Math.E`	Euler'sche Zahl e (~2.718)
`Math.PI`	Kreiszahl π (~3.14159)
`Math.abs(x)`	Betrag der Zahl x
`Math.acos`	arccos(x) mit x ∈ [–1..1]
`Math.asin`	arcsin(x) mit x ∈ [–1..1]
`Math.atan`	arctan(x)
`Math.atan2(x,y)`	Winkel zwischen Ortsvektor (x,y) und der pos. x-Achse
`Math.cos`	cos(x), Winkel x im Bogenmass
`Math.max(x,y)`	Maximum zweier Zahlen x und y
`Math.min(x,y)`	Minimum zweier Zahlen x und y
`Math.pow(x,y)`	Exponentiation x^y
`Math.max(x,y)`	Maximum zweier Zahlen x und y
`Math.random()`	Pseudozufallszahl zwischen 0 und 1
`Math.round(x)`	Runden auf die nächste ganze Zahl
`Math.sin(x)`	sin(x), Winkel x im Bogenmass
`Math.sqrt(x)`	√x, Quadratwurzel
`Math.tan(x)`	tan(x), Winkel x im Bogenmass
`Number.MAX_VALUE`	Grösste darstellbare Zahl (~ 1.79E+308)
`Number.MIN_VALUE`	Kleinste darstellbare Zahl (~ 5E-324)
`isFinite(x)`	Test, ob x nicht NaN oder Infinite ist.
`parseFloat(s)`	Wandelt String s in Zahl oder NaN
`parseInt(s)`	Wandelt String s in ganze Zahl oder NaN