

## Vorlesung



University of Applied Sciences

# DBWP

## Unit Clientseitige Sprachen JavaScript

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

1



University of Applied Sciences



**Prof. Dr. rer. nat.  
Nane Kratzke**  
*Praktische Informatik und  
betriebliche Informationssysteme*

- Raum: 17-0.10
- Tel.: 0451 300 5549
- Email: [kratzke@fh-luebeck.de](mailto:kratzke@fh-luebeck.de)



@NaneKratzke

Updates der Handouts auch über Twitter #dbwp

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

2

## Übergreifende Ziele der Lehrveranstaltung



University of Applied Sciences

Client- und Serverseitige Entwicklung

PHP (Serverseitig)

JavaScript  
(Clientseitig)

„Hosten“ von Apps

Framework Erfahrungen

CMS (Drupal)

WebServices  
(Google-Maps)

jQuery

Datenbank-Integration

Berücksichtigung von Sicherheitsaspekten

HTML-Injections

SQL-Injections

Session Hijacking

Login-Systeme

Um sich weitere Web-Technologien autodidaktisch erarbeiten zu können.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

3

## Units



University of Applied Sciences

Unit 1  
Cloud Computing  
IaaS

Unit 2  
CMS Drupal

Unit 3  
HTML und CSS

Unit 4 - 7  
PHP I - IV

Unit 8  
Sessions, Cookies,  
Formulare und  
Login-System

Unit 9  
JavaScript

Unit 10  
Drupal Module  
Development

Unit 11  
Datenmodellierung

Unit 12 - 13  
Datenbanken und SQL  
Vom Datenmodell zur  
Datenbank

Unit 14  
Datenbank-gestützte  
Web-Anwendungen

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

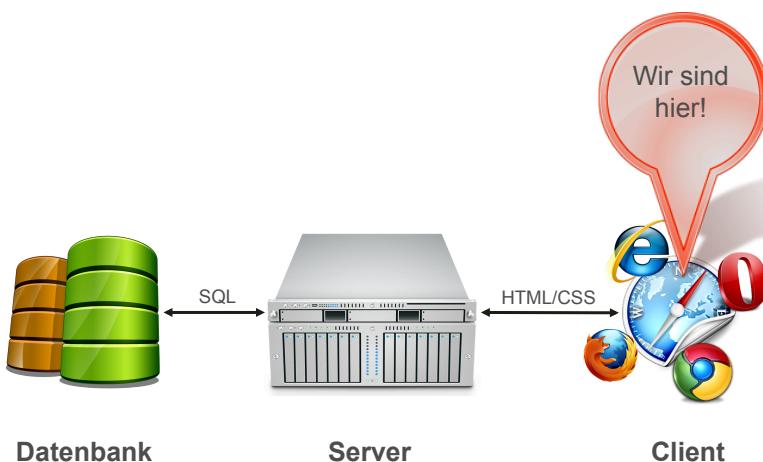
4

## Datenbank – Server – Client

Wo waren wir nochmal?



University of Applied Sciences



Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

5

## Zum Nachlesen ...



University of Applied Sciences



### Kapitel 3

JavaScript einbauen

### Kapitel 4

Programmieren mit JavaScript

### Kapitel 11

Objekte

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

6

## Zum Online Nachlesen ...



University of Applied Sciences



### **SelfHTML (Einführung in JavaScript)**

<http://de.selfhtml.org/javascript/intro.htm>



### **W3Schools.com (JavaScript Tutorial)**

<http://www.w3schools.com/js/>

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

7

## Einordnung der clientseitigen Sprache **JavaScript**



University of Applied Sciences

Programmierparadigma

	Laufzeitmodell		
	Interpreter	Compiler	Byte-Code
Imperativ	X		z.B. C
Funktional	X		
Objekt-orientiert	X		z.B. C++

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

8

## JavaScript



University of Applied Sciences

- JavaScript ist eine clientseitige Programmiersprache.
- JavaScript Programme werden durch einen Webbrowser ausgeführt.
- JavaScript muss hierzu in HTML Dokumenten eingebettet werden.



Der als **ECMA Script (ECMA 262)** standardisierte Sprachkern von JavaScript beschreibt eine dynamisch typisierte, objektorientierte aber klassenlose Skriptsprache. In JavaScript lässt sich sowohl prozedural, funktional als auch objektorientiert programmieren (wenn auch stellenweise unkonventionell, da klassenlos).

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

9

## Hello World!



University of Applied Sciences

```
<html>
  <head>
    <title>Hello World auf JavaScript</title>
  </head>

  <body>
    <script type="text/javascript">
      document.write("Hello World!");
    </script>
  </body>

</html>
```



JavaScript kann mittels SCRIPT Tags in oben gezeigter Form direkt in HTML Dokumente eingebunden werden. Mittels JavaScript kann dann das HTML-Dokument clientseitig verändert werden. Die Funktionsweise demonstriert unten stehender Link.

[http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/HelloWorld/hello\\_world.html](http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/HelloWorld/hello_world.html)

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

10

## Hello World!



University of Applied Sciences

Datei: hello\_world\_external\_js.html

```
<html>
  <head>
    <title>Hello World auf JavaScript</title>
  </head>

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

</html>
```

Datei: hw.js

```
document.write("Hello World! ");
document.write("Dies ist durch ein externes Script erzeugt! ");
```



Insbesondere längere Skripte werden jedoch zumeist in externe Dateien ausgelagert und können wie oben gezeigt inkludiert werden. Die Funktionsweise demonstriert unten stehender Link.

[http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/HelloWorld/hello\\_world\\_external\\_js.html](http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/HelloWorld/hello_world_external_js.html)

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

11

## Hello World!



University of Applied Sciences

Datei: hello\_world\_onload\_js.html

```
<html>
  <head>
    <title>Hello World auf JavaScript</title>
    <script type="text/javascript" src="hw_function.js"></script>
  </head>

  <body onload="hello_world()">
  </body>

</html>
```

Datei: hw\_function.js

```
function hello_world(){
  document.write("Hello World! ");
  document.write("Durch eine externe Funktion erzeugt! ");
}
```



Gebräuchlich ist es ferner, Skripte bereits im Header zu inkludieren und diese dann über Events aufzurufen.

[http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/HelloWorld/hello\\_world\\_onload\\_js.html](http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/HelloWorld/hello_world_onload_js.html)

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

12

## Hello World!



University of Applied Sciences

Datei: hello\_world\_counter.html

```
[...]
<script type="text/javascript" src="hw_counter.js"></script>
[...]

<a href="javascript:hello_world_counter()">Klick mich</a>
<p id="miracle"></p>
```

Datei: hw\_counter.js

```
var klick = 0;

function hello_world_counter() {
    var text = document.createTextNode("Ihr " + ++klick + "ter Klick! ");
    document.getElementById("miracle").appendChild(text);
    br = document.createElement("br");
    document.getElementById("miracle").appendChild(br);
}
```



Auf diese Art und Weise können Sie mittels clientseitig ausgeführten Skripten eine Nutzerinteraktion erzielen, wie dieses Beispiel zeigt.

[http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/HelloWorld/hello\\_world\\_counter.html](http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/HelloWorld/hello_world_counter.html)

Prof. Dr. rer. nat. Nane Kratzke

13

## Programmiersprache JavaScript



University of Applied Sciences

Datentypen und Variablen

Operatorien und Ausdrücke

Routinen und Ablaufsteuerungen

Gängige Datenstrukturen  
(Liste, Mapping, Stack)

Objektorientierung

Sprachbesonderheiten

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

14

## **Variables**

### **Bezeichnungsregeln**



University of Applied Sciences

- Dienen zum Zwischenspeichern von Werten.
- Sind dynamisch typisiert (d.h. der Typ wird aus einem zugewiesenen Wert abgeleitet).
- Können nach folgenden Regeln beliebig benannt werden:
  - Erlaubt sind eine beliebige Folge von Buchstaben (A-Z, a-z), Ziffern (0-9) und dem Unterstrich \_ sowie \$.
  - Es darf keine Ziffer am Anfang stehen.
- Dieselben Benennungsregeln gelten auch für Funktionen!

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

15

## **Gültige und ungültige Bezeichner**



University of Applied Sciences

### **Gültige Bezeichner**

- Galileo\_Press
- GalileoPress
- Galileo2010
- \_Galileo\_
- \$abc

### **Ungültige Bezeichner**

- 1Galileo (beginnt mit meiner Zahl)
- Galileo Press (Leerzeichen)
- Galileo-Press (Bindestrich)
- GalileoPräsenz (Umlaut)

**[A-Z], [a-z], [0-9], \_, \$**

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

16

## Datentypen



University of Applied Sciences

### Numerische Datentypen:

Fließkomma (number), Ganzzahlige Werte (number)

```
var pi = 3.14159265;
var mauerfall = 1989;
var minusHundert = -100;
```

### Datentyp für Wahrheitswerte:

Boolescher Datentyp (boolean)

```
var w = true;
var f = false;
```

### Datentyp für Zeichenketten:

Strings können mittels doppelten Anführungszeichen "" oder einfachen " gekennzeichnet werden.

Mittels des \ können Zeichen escaped werden.

**JavaScript kennt keine Variableninterpolation wie PHP!**

```
var a1 = "Hello World";
var a2 = 'Hello World';

var a3 = 'Hello "Max"';
var a4 = "Hello 'Moritz'";

var a5 = "Hello \"Max\""; 
var a6 = 'Hello \'Moritz\'';
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

17

## Datentypen (hier Strings)



University of Applied Sciences

Der Datentyp String in JavaScript hat – ähnlich wie in JAVA – Objektcharakter, d.h. jeder String trägt Methoden und Attribute „mit sich herum“.

Die wichtigsten sind:

- **length** [die Länge der Zeichenkette]
- **charAt(n)** [das Zeichen an der n. Position der Zeichenkette. Das erste Zeichen hat die Position 0]
- **substring(s, e)** [Liefert eine Teilzeichenkette ab der Position s bis vor dem Zeichen an der Position e]

```
var z = "Web Technologien";
var l = z.length; // z == 16
var e = z.charAt(0); // e == "W"
var w = z.substring(4, z.length); // w == "Technologien"
```

Zeichenketten können ferner mit dem Konkatenationsoperator + verkettet werden.

```
var h = "Hello";
var w = "World";
var hw = h + " " + w; // hw == "Hello World"
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

18

## Deklaration von Variablen

### Schlüsselwort var



University of Applied Sciences

Variablen werden üblicherweise in folgender Form deklariert.

```
var varname = wert;
```

Dies erzeugt eine Variable mit dem Bezeichner **varname**, initialisiert diese und weist den Wert **wert** zu.

Wird diese Variable innerhalb einer Funktion deklariert, so überdeckt sie ggf. eine Variable gleichen Namens mit globaler Gültigkeit. Soll auf eine solche Variable globaler Gültigkeit zugegriffen werden, so darf das Schlüsselwort **var** in diesem Fall nicht verwendet werden.

Variablen können in JavaScript aber auch ohne die Angabe des Schlüsselwortes **var** erzeugt werden. Sie sind dann automatisch global, egal wo sie definiert wurden.

```
varname = wert;
```

**Bevor Variablen in JavaScript nicht über eine der beiden oben genannten Varianten angelegt worden sind, können Sie nicht genutzt werden! Dies ist also anders als in PHP, wo sie notfalls beim ersten Auftreten in einem Ausdruck implizit angelegt werden.**

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

19

## Deklaration von Variablen

### var – globale und lokale Variablen



University of Applied Sciences

```
var a = 0;                      // Globale Variable a
var b = 0;                      // Globale Variable b

function test() {
    a++;                         // Zugriff auf globale Variable a
    var b = 1;                     // Anlegen lokaler Variablen b
                                    // diese über die globale Var. b
    var c = 2;                     // Anlegen lokaler Variablen c
    d = 5;                        // Anlegen GLOBALE Variable d
}

test();

document.writeln("a: " + a);
document.writeln("b: " + b);
document.writeln("d: " + d);
```

Was wird  
ausgegeben?



Oben stehendes Skript zeigt wie JavaScript lokale und globale Variablen im Rahmen der Deklaration handelt.

<http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/VariablenDeklaration/example.html>

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

20

## Dynamische Typisierung von Variablen



University of Applied Sciences

```
function id(x) { return x; }

var v = parseInt("47");
document.writeln("v: " + v + " (" + typeof v + ")");

v = parseFloat("47.11");
document.writeln("v: " + v + " (" + typeof v + ")");

v = "Hello World";
document.writeln("v: " + v + " (" + typeof v + ")");

v = true;
document.writeln("v: " + v + " (" + typeof v + ")");

v = id;
document.writeln("v: " + v + " (" + typeof v + ")");
```

Was wird  
ausgegeben?



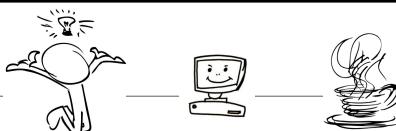
HTTP

<http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/VariablenDeklaration/typing.html>

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

21

## Miniübung:



University of Applied Sciences

Gegeben sei folgender JavaScript Quelltext. Geben Sie bitte alle lokalen und alle globalen Variablen an!

```
var a = 1;
b = 2;

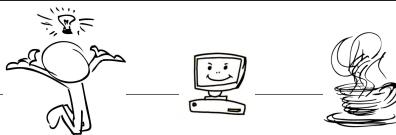
function foo(x, y) {
    c = 0;
    a = x;
    b = y;
    var d = c;
}

function func(i) {
    b = i;
    j = i;
    var h = j;
    return h;
}
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

22

## Miniübung:



Gegeben sei folgender JavaScript Quelltext. Geben Sie bitte die Typen aller globalen Variablen am Programmende an.

```
var a = 1;
b = "Hello World";

function foo(x, y) {
    c = "";
    a = x;
    b = y;
    var d = c;
}

function func(i) {
    b = i;
    j = i;
    var h = j;
    return h;
}

foo(func(true), func(47.11));
```

## Programmiersprache JavaScript

Datentypen und  
Variablen

Operatorien und  
Ausdrücke

Routinen und  
Ablaufsteuerungen

Gängige  
Datenstrukturen  
(Liste, Mapping,  
Stack)

Objektorientierung

Sprachbesonderheiten

## Die wichtigsten Operatorarten



University of Applied Sciences

Arithmetische Operatoren

Relationale Operatoren

Logische Operatoren

Bedingte Auswertung

Zuweisungsoperatoren

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

25

## Liste arithmetischer Operatoren (JavaScript)



University of Applied Sciences

Operator	Bezeichnung	Bedeutung
<code>++</code>	Pre-Inkrement Post-Inkrement	<code>a++</code> = Erhöhung von a um 1 nach Auswertung eines Ausdrucks. <code>++a</code> = Erhöhung von a um 1 vor Auswertung eines Ausdrucks
<code>--</code>	Pre-Dekrement Post-Dekrement	<code>a--</code> = Reduktion von a um 1 nach Auswertung eines Ausdrucks. <code>--a</code> = Reduktion von a um 1 vor Auswertung eines Ausdrucks
<code>+</code>	Summe	<code>a + b</code> = Summe von a und b
<code>-</code>	Subtraktion	<code>a - b</code> = Differenz von a und b
<code>*</code>	Multiplikation	<code>a * b</code> = Produkt von a und b
<code>/</code>	Division	<code>a / b</code> = Quotient von a und b.
<code>%</code>	Restwert (Modulo)	<code>a % b</code> = Rest der ganzzahligen Division von a durch b.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

26

## Liste relationaler Operatoren (JavaScript)



University of Applied Sciences

Operator	Bezeichnung	Bedeutung
<code>==</code>	gleich	a == b ergibt true wenn a und b denselben Wert haben.
<code>===</code>	identisch	a === b ergibt true, wenn a == b und typeof a == typeof b.
<code>!=</code>	ungleich	a != b ergibt true, wenn a und b nicht denselben Wert haben.
<code>!==</code>	nicht identisch	a !== b ergibt true, wenn a und b nicht denselben Wert oder denselben Type haben.
<code>&lt;</code>	kleiner	a < b ergibt true wenn der Wert von a kleiner als der Wert von b ist.
<code>&gt;</code>	größer	a > b ergibt true wenn der Wert von a größer als der Wert von b ist.
<code>&lt;=</code>	Kleiner gleich	a <= b ergibt true wenn der Wert von a kleiner als der Wert von b oder gleich dem Wert von b ist.
<code>&gt;=</code>	größer gleich	a >= b ergibt true wenn der Wert von a größer als b oder gleich dem Wert von b ist.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

27

## Liste logischer Operatoren (JavaScript)



University of Applied Sciences

Operator	Bezeichnung	Bedeutung
<code>!</code>	Logisches NICHT	!a ergibt true wenn der Wert von a false ist und false, wenn der Wert von a true ist.
<code>&amp;&amp;</code>	UND	a && b ergibt true, wenn der Wert von a und der Wert von b true sind. Ist bereits a false wird b nicht mehr ausgewertet (Short Circuit Verhalten)
<code>  </code>	ODER	a    b ergibt true, wenn mindestens einer der beiden Ausdrücke a oder b zu true ausgewertet werden kann. Ist bereits a true wird b nicht mehr ausgewertet. (Short Circuit Verhalten)

JavaScript kennt kein logisches XOR.

A “xor“ B  $\Leftrightarrow$  (A == !B) || (!A == B)

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

28

## **Bedingte Auswertung (JavaScript)**

**a ? b : c**



University of Applied Sciences

- Der Fragezeichen Operator ?: ist der einzige dreiwertige Operator
- Kann häufig eingesetzt werden, um if-Abfragen zu vermeiden.
- a ? b : c
  - Ist a true wird b zurückgeliefert
  - Ist a false wird c zurückgeliefert

Die **Bedingte Auswertung** ist ein Ausdruck, keine Kontrollstruktur  
(if-Anweisung)!

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

29

## **Liste der Zuweisungsoperatoren (JavaScript)**



University of Applied Sciences

Operator	Bezeichnung	Bedeutung
=	Einfache Zuweisung	$a = b$ weist a den Wert von b.
+=	Additionszuweisung	$a += b$ weist a den Wert von $a + b$ zu. $a += b$ identisch zu $a = a + b$
-=	Subtraktionszuweisung	Analog += Operator mit -
*=	Multiplikationszuweisung	Analog += Operator mit *
/=	Divisionszuweisung	Analog += Operator mit /
%=	Modulozuweisung	Analog += Operator mit %

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

30

## Vorrangregeln Operatorgruppen



University of Applied Sciences

Operatoren (JS)	Operatoren (PHP)	Bezeichnung
<code>++, --, !</code>	<code>++, --, !</code>	Inkrement, Dekrement, Nicht
<code>*, /, %</code>	<code>*, /, %</code>	Multiplikation, Division, Modulo
<code>+, -</code>	<code>+, -</code>	Addition, Subtraktion, String-Konkatenation
<code>&lt;, &lt;=, &gt;, &gt;=</code>	<code>&lt;, &lt;=, &gt;, &gt;=</code>	Kleiner, Kleiner-Gleich, Größer, Größer-Gleich
<code>==, !=, ===, !==</code>	<code>==, !=, ===, !==</code>	Gleich, Ungleich, Identität, Nicht-Identität
<code>&amp;&amp;</code>	<code>&amp;&amp;</code>	Logisches Und
<code>  </code>	<code>  </code>	Logisches Oder
<code>?:</code>	<code>?:</code>	Bedingte Auswertung
<code>=, +=, -=, *=, ...</code>	<code>=, +=, -=, *=, ...</code>	Zuweisungen
unbekannt	xor	Exklusiv Oder

Operatoren in JavaScript und PHP im wesentlichen identisch.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

31

## Programmiersprache JavaScript



University of Applied Sciences

Datentypen und Variablen

Operatoren und Ausdrücke

Routinen und Ablaufsteuerungen

Gängige Datenstrukturen  
(Liste, Mapping, Stack)

Objektorientierung

Sprachbesonderheiten

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

32

## Klassische Datenstrukturen



The diagram features three circular segments arranged in a triangle. The top-left segment contains the word 'Map', the top-right segment contains 'List', and the bottom segment contains 'Stack'. To the left of the diagram is a photograph of a wooden filing cabinet with many drawers, some of which are open showing cards. To the right is a 3D white humanoid figure holding a long red ribbon that forms a loop, symbolizing a list or stack.

FACH  
HOCHSCHULE  
LÜBECK  
University of Applied Sciences

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

33

## JavaScript kennt hierzu Arrays und Objekte



Die fundamentalen Datenstrukturen List und Stack werden in JavaScript durch den Datentyp Array abgebildet.

Die fundamentalen Datenstrukturen Map (Key Value Paare) können in JavaScript über Objekte und deren Attribute abgebildet werden.

FACH  
HOCHSCHULE  
LÜBECK  
University of Applied Sciences

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

34

## Erzeugung von Arrays und Zugriff auf Elemente des Arrays



University of Applied Sciences

Arrays werden wie folgt erzeugt.

```
var a = new Array("Hello", " ", "World");
```

Auf die Elemente innerhalb des Arrays kann über deren Index zugegriffen werden, der in eckigen Klammern notiert wird:

```
document.writeln(a[0]);  
document.writeln(a[1]);  
document.writeln(a[2]);
```

Hello

Leerzeichen

World

Arrays können auch in folgender Kurznotation erzeugt werden.

```
var a = ["Hello", " ", "World"];
```

*Hinweis:* Diese Notation wird in dieser Vorlesung bevorzugt.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

35

## Die wichtigsten Attribute und Methoden von Array Objekten (I)



University of Applied Sciences

Anzahl der Elemente in einem Array

```
var a = ["Hello", " ", "World"]  
a.length == 3
```

Verwandelt einen Array in eine Zeichenkette. Parameter: Trennzeichenkette  
Rückgabe: Zeichenkette

```
var a = [1, 2, 3];  
a.join("+") == "1+2+3"
```

Entfernt das letzte Element aus einem Array. Rückgabe: Entferntes Element

```
var a = [1, 2, 3];  
(a.pop() == 3) && (a == [1, 2])
```

Hängt Elemente an. Parameter: ein oder mehrere anzuhängende Elemente.  
Rückgabe: Länge des Arrays.

```
var a = [1, 2, 3];  
(a.push(4,5) == 5) && (a == [1,2,3,4,5])
```

Kehrt die Elementreihenfolge innerhalb eines Arrays um. Rückgabe: Keine

```
var a = [1, 2, 3];  
a.reverse() => a == [3,2,1]
```

Fügt Arrays zusammen. Parameter: zu verknüpfende Arrays. Rückgabe:  
zusammengefügtes Array

```
var a = [1, 2, 3]; var b = [4, 5, 6]  
a.concat(b) == [1,2,3,4,5,6]
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

36

## Die wichtigsten Attribute und Methoden von Array Objekten (II)



University of Applied Sciences

Entfernt das erste Element aus einem Array.  
Rückgabe: Entferntes Element

```
var a = [1, 2, 3];
(a.shift() == 1) && (a == [2, 3])
```

Fügt am Anfang eines Arrays ein oder mehrere neue Elemente ein. Parameter: einzufügenden Elemente, Rückgabe: Länge des Arrays

```
var a = [3, 4, 5]
(a.unshift(2) == 4) && (a == [2, 3, 4, 5])
```

Extrahiert einen Teil aus einem Array. Parameter:  
Indexnummer des ersten zu extrahierenden Elements,  
Indexnummer des Elements, dass dem letzten zu  
extrahierenden Element folgt. Rückgabe:  
Extrahierte Elemente als Array zurück. Wird der zweite  
Parameter weggelassen, werden alle Elemente bis um  
Ende des Arrays extrahiert.

```
var a = [5, 4, 3, 1, 2];
a.slice(1) == [4, 3, 1, 2]
a.slice(1, 3) == [4, 3]
```

Sortiert die Elemente eines Arrays.  
Wenn Sie keinen Parameter  
übergeben, wird lexikalisch sortiert,  
numerische Werte werden also  
intern in Zeichenketten verwandelt  
und wie Zeichenketten sortiert.

```
// Lexikographisch Sortierten
var a = ["World", "Hello"];
a.sort() => a == ["Hello", "World"]
```

```
// Numerisch sortieren
var a = [5, 4, 3, 1, 2];
a.sort(function(x,y) {return x-y}) => a == [1, 2, 3, 4, 5]
```

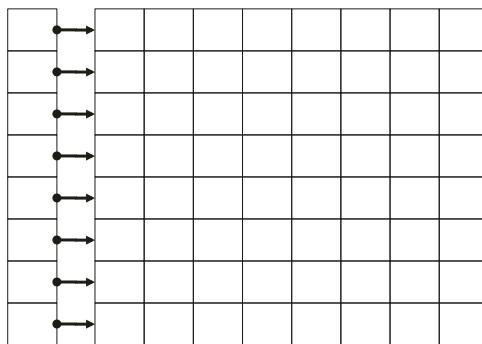
Prof. Dr. rer. nat. Nane Kratzke | 37  
Praktische Informatik und betriebliche Informationssysteme

## Mehrdimensionale Arrays



University of Applied Sciences

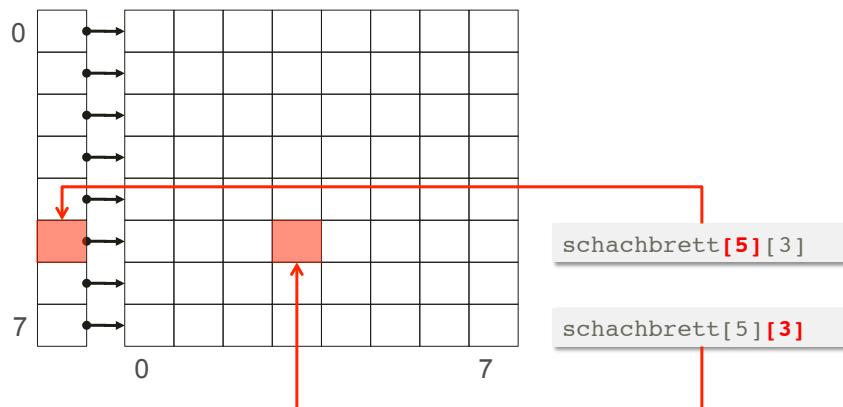
Mehrdimensionale Arrays werden als Arrays von Arrays angelegt.



Prof. Dr. rer. nat. Nane Kratzke | 38  
Praktische Informatik und betriebliche Informationssysteme

## Mehrdimensionale Arrays

Mehrdimensionale Arrays werden als Arrays von Arrays angelegt.

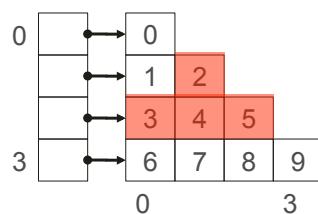


Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme 39

## Mehrdimensionale Arrays

Es ist auch möglich nicht rechteckige Arrays anzulegen.

```
var xs = [  
  [0],  
  [1, 2],  
  [3, 4, 5],  
  [6, 7, 8, 9]  
];
```



Welchen Wert hat  
dieser Ausdruck?

a[1][1] = ???

a[2] = ???

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme 40

## Erzeugung von Mappings Key Value Paare



University of Applied Sciences

Als assoziative Arrays (Mappings) bezeichnet man Sammlungen von Elementen, bei denen der Zugriff auf einzelne Elemente (Value) mit Hilfe einer Zeichenkette oder sonstigen Objekts (Key) erfolgt. Im Gegensatz zu anderen Programmiersprachen gibt es in JavaScript keine assoziativen Arrays (Mappings).

Arrays in JavaScript erlauben nur den Zugriff auf die Elemente über Indexnummern.

Man kann jedoch mit Hilfe von `Object()` das Verhalten eines assoziativen Arrays teilweise nachbauen (sofern man als Schlüssel nur Zeichenketten benötigt, ggf. muss man komplexere Objekte in Zeichenketten serialisieren).

Es handelt sich dann um ein allgemeines Objekt, an dem die Elemente als Objekteigenschaften hängen. Auf Eigenschaften kann man nicht nur über den Punkt-Operator zugreifen (`objekt.eigenschaft`), sondern auch über eine Notation, die dem Index-Operator bei Arrays gleicht (`objekt["eigenschaft"]`).

Dadurch lassen sich beliebige Schlüssel erzeugen. Das bringt einige Fallstricke mit sich. Die Elemente eines solchen "Mappings" können stets nur mit dem Namen oder über eine `for-in`-Schleife angesprochen werden.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

41

## Beispiel: Erzeugung von Mappings Key Value Paare



University of Applied Sciences

Anlegen eines Objekts inkl. beschreibender Attribute (Punkt Notation) ...

```
var person = new Object();
person.vorname = "Max";
person.nachname = "Mustermann";
```

... ist absolut gleichwertig zu dieser Index Operator Form.

```
var person = new Object();
person["vorname"] = "Max";
person["nachname"] = "Mustermann";
```

Dies gilt natürlich auch für den lesenden Zugriff (beide Formen sind absolut gleichwertig):

```
document.writeln("Vorname: " + person.vorname);
document.writeln("Nachname: " + person.nachname);
```

Vorname: Max  
Nachname: Mustermann

```
document.writeln("Vorname: " + person["vorname"]);
document.writeln("Nachname: " + person["nachname"]);
```

Vorname: Max  
Nachname: Mustermann

Das „Mapping“ kann auch mit folgender Kontrollstruktur durchlaufen werden:

```
for (var key in person) {
    document.writeln(key + ": " + person[key]);
}
```

Vorname: Max  
Nachname: Mustermann

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

42

## Programmiersprache JavaScript



University of Applied Sciences

Datentypen und Variablen

Operatoren und Ausdrücke

Routinen und Ablaufsteuerungen

Gängige Datenstrukturen  
(Liste, Mapping, Stack)

Objektorientierung

Sprachbesonderheiten

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

43

## Anweisungen (; or not ;)



University of Applied Sciences

Imperative Programme sind durch die sequentielle Abarbeitung von Anweisungen gekennzeichnet. Üblicherweise nutzen Programmiersprachen daher ein Trennzeichen, um zwei Anweisungen von einander zu trennen.

Bei vielen Programmiersprachen ist dies „;“. So auch bei JavaScript. In JavaScript ist es jedoch auch möglich zwei Anweisungen durch einen Zeilenumbruch voneinander zu trennen. D.h. die beiden nachfolgenden Codebeispiele, sind für den Interpreter ein und dieselbe Anweisungsabfolge. Beides ist korrektes JavaScript.

```
var person = new Object();
person.vorname = "Max";
person.nachname = "Mustermann";
```

```
var person = new Object()
person.vorname = "Max"
person.nachname = "Mustermann"
```

In JavaScript ist der Einsatz von ; also nicht zwingend erforderlich. Insbesondere im Bereich der funktionalen Programmierung wird in dieser Vorlesung gerne darauf verzichtet.

Die sequentielle Abarbeitung von Anweisungen kann dabei durch

- **Funktionen** (Routinen, Methoden) sowie
- **Ablaufsteuerungen** (Schleifen, Verzweigungen, Exception Handling, etc.) beeinflusst werden. Dies ist Gegenstand dieses Abschnitts.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

44

## Funktionen



University of Applied Sciences

### Syntax:

```
function Name([Parameter]) {  
    // Anweisungen der Function  
    return ausdruck; // ggf.  
}
```

- Codefragmente,
- die Anweisungen kapseln
- um eine ggf. parametrisierte Funktionalität
- zu realisieren
- und ggf. einen Rückgabewert haben.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

45

## Funktionen Beispiel



University of Applied Sciences

```
function conc(w1, w2, w3) {  
    var concstr = w1 + w2 + w3;  
    return concstr;  
}  
  
document.writeln(conc("Hello", " ", "World"));
```

Funktionsname

Aufrufparameter

Rückgabe

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

46

## Funktionen Beispiel



University of Applied Sciences

```
function conc(w1, w2, w3) {  
    var concstr = w1 + w2 + w3;  
    return concstr;  
}  
  
document.writeln(conc("Hello", " ", "World"));
```

Funktionen-  
name

Aufruf-  
parameter

Rückgabe

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

47

## Funktionen Beispiel



University of Applied Sciences

```
function conc(w1, w2, w3) {  
    var concstr = w1 + w2 + w3;  
    return concstr;  
}  
  
document.writeln(conc("Hello", " ", "World"));
```

Funktionen-  
name

Aufruf-  
parameter

Rückgabe

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

48

## Funktionen Beispiel



University of Applied Sciences

```
function conc(w1, w2, w3) {  
    var concstr = w1 + w2 + w3;  
    return concstr;  
}  
  
document.writeln(conc("Hello", " ", "World"));
```

Methoden-  
name

Aufruf-  
parameter

Rückgabe

**Merk:** Funktionen müssen kein return Statement in JavaScript beinhalten. Sie liefern dennoch in diesen Fällen immer den Rückgabewert undefined zurück.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

49

## Funktionen „Default Parameter“



University of Applied Sciences

**Merk:** In JavaScript definierte Funktionen müssen nicht mit der Anzahl an Parametern aufgerufen werden, mit der sie definiert wurden.

Parameter, die in der Funktionssignatur definiert wurden, aber in einem Funktionsaufruf nicht übergeben wurden, werden mit dem Wert undefined belegt.

Parameter, die einer Funktion übergeben werden, aber nicht in der Funktionssignatur definiert wurden, werden ignoriert.

```
function conc(w1, w2, w3) {  
    var concstr = w1 + " " + w2 + " " + w3;  
    return concstr;  
}  
  
document.writeln(conc("A", " B ", "C", "D") + "<br>");  
document.writeln(conc("A", " B ", "C") + "<br>");  
document.writeln(conc("A", " B ") + "<br>");  
document.writeln(conc("A") + "<br>");  
document.writeln(conc() + "<br>");
```

Was wird  
ausgegeben?



HTTP

<http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/Functions/undefined.html>

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

50

## Funktionen

### Call by reference vs. Call by value



University of Applied Sciences

- Es gibt in gängigen Programmiersprachen grundsätzlich zwei Arten Parameter an eine Routine zu übergeben

#### Call by reference

- Es wird ein Zeiger auf eine Variable übergeben.
- Innerhalb der Routine wird über den Zeiger auf der Variable außerhalb der Routine gearbeitet.
- **Der Inhalt der Variable außerhalb der Routine wird verändert.**

#### Call by value

- Der Wert einer Variable wird in die Parametervariable kopiert.
- Innerhalb der Routine wird auf der Kopie gearbeitet.
- **Der Inhalt der Variable außerhalb der Routine wird nicht verändert.**

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

51

## Funktionen

### Trickreiches Call by Value (I)



University of Applied Sciences

**Merke:** JavaScript folgt dem Call by Value Verfahren bei Parameterübergabe an Funktionen. Damit können Änderungen an Parametern innerhalb einer Funktion (**eigentlich!!!**) nicht nach außen durchschlagen.

**Doch Vorsicht:** Es gibt zwei Arten von Datentypen in JavaScript (ähnlich wie JAVA).

- Number, Float, Boolean und String werden nach **Wertsemantik** gespeichert, d.h. in einer Variablen steht der Wert dieser Datentypen.
- Alle anderen Datentypen Arrays, Objekte werden nach der **Referenzsemantik** gespeichert, d.h. in einer Variablen steht die Referenz auf ein Objekt.

Auch Referenzen werden in JavaScript nach dem Call by Value Verfahren übergeben.

- Werden also nur die Referenzen in einer Funktion geändert, schlägt dies nicht nach außen durch.
- **Wird über die Referenz allerdings etwas am Objekt geändert, so schlägt dies natürlich nach außen durch.**

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

52

## Funktionen

### Trickreiches Call by Value (II)



University of Applied Sciences

Dies führt zu folgendem Verhalten:

```
function swap(x1, x2) {  
    var dummy = x1;  
    x1 = x2; x2 = dummy;  
}
```

```
function swap(xs) {  
    var dummy = xs[0];  
    xs[0] = xs[1]; xs[1] = dummy;  
}
```

Es werden die Werte von Parametern mit Wertesemantik verändert:

```
var s1 = "Hello"; var s2 = "World";  
swap(s1, s2);  
document.writeln(s1 + " " + s2);
```

Hello World  
schlägt nicht durch

Es werden die Referenzen von Parametern mit Referenzsemantik verändert:

```
var S1 = new String("HELLO"); var S2 = new String("WORLD");  
swap(S1, S2);  
document.writeln(S1 + " " + S2);
```

HELLO WORLD  
schlägt nicht durch

Es werden die Objektbestandteile von Parametern mit Referenzsemantik über die Referenz verändert:

```
var strs = ["Hello", "World"];  
swap(strs);  
document.writeln(strs[0] + " " + strs[1]);
```

World Hello  
schlägt durch

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

53

## Anonyme Funktionen (Lambda Funktionen)



University of Applied Sciences

Funktionen sind Elemente erster Ordnung

- Sie können Variablen zugewiesen werden.
- Sie können adhoc und unbenannt definiert werden.
- Sie können anderen Funktionen als Wert für einen Parameter übergeben werden.

Sprachmittel

- der sogenannten funktionalen Programmierung,
- die häufig als zu akademisch abgestempelt werden,
- jedoch seit geraumer Zeit in alle modernen Sprachen Einzug halten.

Beispiel:

```
var lambda = function(a) { return a/2; };  
document.writeln(lambda(4));
```

2

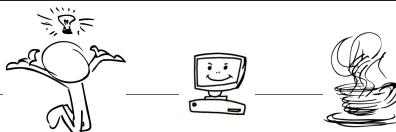
```
var list = [6, 8, 10, 14, 18];  
var map = function(f, xs) {  
    return (xs.length == 0) ? [] :  
        [f(xs[0])].concat(map(f, xs.slice(1)))  
}  
document.writeln(map(lambda, list).join(", "));
```

3,4,5,7,9

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

54

## Miniübung:



Welche Ausgaben produzieren die folgenden auf Lambda Funktionen (anonymen Funktionen) beruhenden Anweisungen?

```
var f = function(a,b) { return a + b };
document.writeln(f(5,6));

var g = function(f, a) { return f(a) };
document.writeln(g(function (x) { return 2*x }, 10));

var q = function(a) { return a*a };

var map = function(f, xs) {
  if (xs.length == 0) return [];
  return [f(xs[0])].concat(map(f, xs.slice(1)));
}

var mapper = function(f, xs, s) {
  return map(f, xs).join(s);
}

document.writeln(mapper(q, [3, 4, 5, 6], ", "));
```



HTTP  
<http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/javascript/Functions/lambda.html>

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

55

## Frage: Ist JavaScript eine funktionale Programmiersprache?

- First Class Object Check?
- Higher Order Function Check?
- List and List Operator Check?  
(`first`, `rest`, `empty`,  
`single`)
- Typical Function Operator Check?  
(`map`, `reduce`, `filter`)



Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

56

## Frage: Ist JavaScript eine funktionale Programmiersprache?



University of Applied Sciences

- First Class Object Check?
  - Sind Funktionen Objekte erster Klasse?
  - Können anonyme Funktionen definiert werden?

Funktionen sind in JavaScript Objekte erster Klasse. Funktionen können z.B. Variablen als „Wert“ zugewiesen werden.

```
function id(x) { return x }  
var f = id;
```

In JavaScript können auch anonyme Funktionen definiert werden.

```
var d = function (x) { return 2*x }
```



Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

57

## Frage: Ist JavaScript eine funktionale Programmiersprache?



University of Applied Sciences

- Higher Order Function Check?

Funktionen können anderen Funktionen als Parameter übergeben werden. Hier am Beispiel der Identitätsfunktion veranschaulicht.

```
function id(x) { return x }  
var g = id(id);
```



Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

58

## Frage: Ist JavaScript eine funktionale Programmiersprache?



University of Applied Sciences

- List and List Operator Check?  
(first, rest, notEmpty, single)

Es können Listen in JavaScript angelegt werden.

```
var list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
```



Funktionen zum Zugriff auf das erste Element einer Liste und den Rest einer Liste:

```
function first(xs) { return xs[0] }
function rest(xs) { return xs.slice(1) }
```

Funktionen zum Prüfen, ob eine Liste leer ist oder nur aus einem Element besteht:

```
function notEmpty(xs) { return xs.length > 0 }
function single(xs) { return xs.length == 1 }
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

59

## Frage: Ist JavaScript eine funktionale Programmiersprache?



University of Applied Sciences

- Typical Function Operator Check?  
(map, reduce, filter)

map wendet eine Funktion f auf jedes Element einer Liste an und liefert eine Liste.

```
function map(f, xs) {
  return !notEmpty(xs) ? [] : [f(first(xs))].concat(map(f, rest(xs)));
}
```

reduce verknüpft alle Elemente einer Liste mit einer zweiwertigen Funktion f.

```
function reduce(f, xs) {
  return single(xs) ? first(xs) : f(first(xs), reduce(f, rest(xs)));
}
```

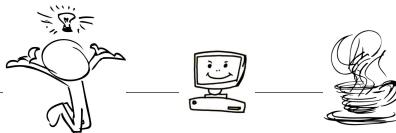
filter erzeugt eine Liste mit allen Elementen einer Liste, die einer booleschen Filterfunktion c genügen.

```
function filter(c, xs) {
  if (!notEmpty(xs)) return [];
  return (c(first(xs)) ? [first(xs)] : []).concat(filter(c, rest(xs)));
}
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

60

## Miniübung:



Welche Ausgaben produzieren die folgenden auf klassischen funktionalen Listenfunktionen beruhenden Anweisungen?

```
function add(x, y) { return x + y }
function double(x) { return 2 * x }
document.writeln(reduce(add, map(double, [1, 2, 3, 4, 5])));
```

30

```
function even(x) { return x % 2 == 0 }
function triple(x) { return 3 * x }
function conc(s1, s2) { return s1 + "++" + s2 }
document.writeln(reduce(conc, filter(even, map(triple, [1, 2, 3, 4, 5]))));
```

6-+12



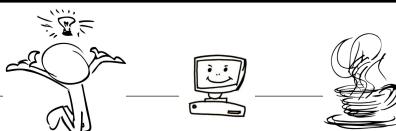
Den gesamten „Funktionale Feature Check“ und dieses Beispiel können Sie hier noch einmal nachvollziehen:

<http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/FunktionaleFeatures/funktional.html>

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

61

## Miniübung:



Es ist Ihnen folgende Array Struktur gegeben:

Sie sollen nun eine rein funktionale Lösung entwickeln, um daraus eine HTML Tabelle zu erzeugen.

```
var xs = [
  [1, 2, 3, 4],
  [5, 6, 7],
  [8, 9],
  [0]
];
```

**Tipp:** Definieren Sie sich Funktionen, um ein Tabellenelement, eine Tabellenzeile und die ganze Tabelle zu erzeugen. Die Funktionen dürfen aufeinander aufbauen und die gerade eben eingeführten Funktionen (map, reduce, etc.) nutzen. Sie dürfen keine Schleifen oder Anweisungsfolgen nutzen!



Die Lösung können Sie hier noch einmal nachvollziehen:

<http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/FunktionaleFeatures/table.html>

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

62

## Ablaufsteuerung



University of Applied Sciences

### Verzweigungen

- if
- Switch
- Exception Handling

### Schleifen

- while
- do
- for
- foreach

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

63

### Verzweigungen if-Abfrage



University of Applied Sciences

- Verzweigungen dienen dazu bestimmte Programmteile nur beim Eintreten vorgegebener Bedingungen auszuführen.

If Variante

```
if (ausdruck)
    anweisung;
```

If Else Variante

```
if (ausdruck)
    anweisung;
else
    anweisung;
```

If Block Variante

```
if (ausdruck) {
    Block von Anweisungen;
}
```

If Else Block Variante

```
if (ausdruck) {
    Block von Anweisungen;
} else {
    Block von Anweisungen;
}
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

64

## Verzweigungen Switch-Anweisung



University of Applied Sciences

- Switch Anweisung ist eine Mehrfachverzweigung.
- sie wertet einen Ausdruck aus
- und springt einen case Zweig oder den default Zweig an.

### Syntax:

```
switch (ausdruck) {  
    case Wert: anweisung;  
    ...  
    default: anweisung;  
}
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

65

## Verzweigungen Switch-Anweisung (Beispiel)



University of Applied Sciences

Welche Ausgabe erzeugt dieser Code?

```
var i = 4;  
  
switch (i % 3) {  
    case 1: echo "Rest 1";  
    case 2: echo "Rest 2";  
    case 3: echo "Rest 3";  
    default: echo "Rest 0";  
}
```

- |        |   |
|--------|---|
| Rest 1 | Achtung: Nachdem ein case- oder default-Label angesprungen wurde, werden alle dahinter stehenden Anweisungen ausgeführt.  |
| Rest 2 |   |
| Rest 3 |   |
| Rest 0 | Will man das nicht, muss man das Label mit einer break Anweisung dazu zwingen, am Ende der switch-Anweisung fortzusetzen. |

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

66

## Verzweigungen Switch-Anweisung (Beispiel)



University of Applied Sciences

Welche Ausgabe erzeugt dieser Code?

```
vari = 4;

switch (i % 3) {
    case 1: echo "Rest 1"; break;
    case 2: echo "Rest 2"; break;
    case 3: echo "Rest 3"; break;
    default: echo "Rest 0";
}
```

Rest 1

Die ergänzende break Anweisung realisiert die Semantik der switch Anweisung, wie man sie intuitiv erwarten würde.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

67

## Exception Handling



University of Applied Sciences

JavaScript bietet auch ein Exception Handling mittels try catch Blöcken an. Das Prinzip funktioniert wie bei bspw. JAVA. Ist allerdings nicht ganz so mächtig – es werden bspw. Nicht mehrere catch Blöcke zum zielgerichteten Exception Handling angeboten. Es gibt auch keine finally Blöcke etc.

Syntax:

```
try {
    // Code unter Exception Handling
} catch (ex) {
    // Code zur Ausnahmebehandlung
}
```

Mittels einer throw Anweisung können eigene Exceptions ausgelöst werden.

Exception Handling dient dazu Logik Code und Fehlerbehandlungscode besser voneinander trennen zu können und so besser lesbaren Code zu erzeugen.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

68

## Exception Handling (Beispiel)



University of Applied Sciences

```
var op = function (x, y) { return x / y; };

try {
    for (var i = 0; i <= 3; i++) {
        for (var j = 3; j >= 0; j--) {
            document.writeln(i + " / " + j + " == " + op(i,j) + "<br>";
        }
    }
} catch(ex) {
    window.alert("Es ist die Exception " + ex + " aufgetreten");
}

try {
    for (var i = 0; i <= 3; i++) {
        for (var j = 3; j >= 0; j--) {
            if (j == 0) throw "Durch Null teilen!";
            document.writeln(i + " / " + j + " == " + op(i,j) + "<br>";
        }
    }
} catch(ex) {
    window.alert("Es ist die Exception " + ex + " aufgetreten");
}
```

Was wird  
ausgegeben?

<http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/ExceptionHandling/ex.html>

**Merk:** In JavaScript kann man offenbar durch 0 teilen.  
0 / 0 = NaN (Not a number); x / 0 = Infinity (wenn x > 0)

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

69

## Ablaufsteuerungen



University of Applied Sciences

### Verzweigungen

- if
- switch
- exception Handling

### Schleifen

- while
- do
- for
- for in

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

70

## Abweisende Schleife



University of Applied Sciences

### Syntax:

```
while (ausdruck) {  
    anweisung;  
}
```

- Prüfen des Ausdrucks
- Solange dieser True ist, wird der Anweisungsblock oder Einzelanweisung ausgeführt
- Ist der Ausdruck bereits zu Beginn false wird der Anweisungsblock nicht ausgeführt, daher **abweisende Schleife**.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

71

## Nicht abweisende Schleife



University of Applied Sciences

### Syntax:

```
do {  
    anweisung;  
} while (ausdruck);
```

- Der Anweisungsblock wird mindestens einmal ausgeführt, daher **nicht abweisende Schleife**.
- Prüfen des Ausdrucks
  - Ist dieser True, wird der Anweisungsblock oder Einzelanweisung wieder ausgeführt
  - Ist dieser false wird die Programmausführung hinter dem while Ausdruck fortgesetzt.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

72

## Zählschleife for – klassische Variante



University of Applied Sciences

### Syntax:

```
for (init; test; update) {  
    anweisung;  
}
```

#### Init Ausdruck

- Aufruf einmalig **vor Start** der Schleife
- Optional
- mehrere kommasseparierte Ausdrücke mögl.
- Variablen-deklaration möglich

#### Test Ausdruck

- Aufruf **jew. am Anfang** einer Schleife
- Optional, wenn nicht angegeben wird true gesetzt
- Schleife wird nur ausgeführt, wenn Ausdruck true

#### Update Ausdruck

- Aufruf **jew. am Ende** der Schleife
- Optional
- Mehrere kommasseparierte Ausdrücke möglich
- Dient dazu den Schleifenzähler zu ändern

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

73

## Zählschleife for – klassische Variante (Beispiele)



University of Applied Sciences

```
for (var i = 1; i <= 3; i++) { document.writeln(i); };
```

```
var i = 1;  
for (;;) {  
    if (!(i <= 3)) break;  
    document.writeln(i);  
    i++;  
}
```

Ist das eine  
syntaktisch korrekte  
for Schleife?

```
//init-Anweisung(en) – einmalig  
//test-Ausdruck – vor Schleifenlauf  
//Update-Anweisung(en) – am Ende
```

Jede for-Schleife kann nach diesem  
Muster umformuliert werden.



<http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/Schleifen/for.html>

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

74

## Zählschleife for in Variante



University of Applied Sciences

### Syntax:

```
for([var] v in obj) {  
    anweisung;  
}
```

- **v** ist die „Laufvariable“
- **obj** ist ein Array oder ein Object.
- Zu lesen ist dieser Ausdruck **for each v in obj**, d.h.
  - es werden alle Bestandteile aus einem Array (Einträge) bzw. einem Objekt (Attribute) durchlaufen und jeweils pro Schleifendurchlauf der Laufvariablen **v** zugewiesen.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

75

## Beispiele: for in Schleifen



University of Applied Sciences

### Variante zum Durchlaufen eines Arrays:

```
var as = [7, 5, 3, 1];  
for (var i in as) { document.writeln(as[i] + "<br>"); }
```

### Variante zum Durchlaufen aller Attribute eines Objekts/Mappings (wenn ein Objekt als Map genutzt wird):

```
var obj = new Object();  
obj.attr1 = "Wert 1";  
obj.attr2 = "Wert 2";  
obj.meth = function (x) { return x } ;  
  
for (var x in obj) {  
    document.writeln(x + ":" + obj[x] + "<br>");  
}
```

Was wird in beiden  
Fällen  
ausgegeben?

<http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/Schleifen/forin.html>



Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

76

## Programmiersprache JavaScript



University of Applied Sciences

Datentypen und Variablen

Operatoren und Ausdrücke

Routinen und Ablaufsteuerungen

Gängige Datenstrukturen  
(Liste, Mapping, Stack)

Objektorientierung

Sprachbesonderheiten

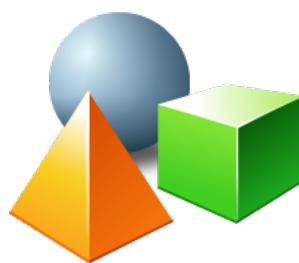
Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

77

## Wann ist eine Sprache objektorientiert?



University of Applied Sciences



**JavaScript kennt keine Klassen!**

**Ist die Sprache dennoch objektorientiert?**

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

78

## Wann ist eine Sprache objektorientiert?



University of Applied Sciences

Bei einer Präsentation bei Apple Mitte der 80er hielt ein Mitarbeiter einen Vortrag über die neue „objektorientierte“ Programmiersprache Oberon. Da meldete sich Alan Kay zu Wort:



Der Vortragende meinte darauf: „Nun, wer kann schon genau sagen, was objektorientiert ist und was nicht.“ Woraufhin Alan Kay zurückgab: „Ich kann das, ich bin Alan Kay und ich habe den Begriff geprägt.“

*Hinweis: Alan Kay gilt als der Erfinder von SmallTalk, der ersten konsequent objektorientierten Sprache weltweit.*

*In Anlehnung an: Lahres, Rayman – Objektorientierte Programmierung*

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

79

## Wann ist eine Sprache objektorientiert? Vier Kontrollfragen



University of Applied Sciences

- Können Objekte angelegt und diesen Attribute und Methoden zugewiesen werden?
- Können Objekte voneinander abgeleitet werden, kann das Sohnobjekt auf die Attribute und Methoden des Vaters zugreifen? (Vererbung)
- Ist ein Sohnobjekt immer auch vom Typ des Vaters? (Polymorphie)
- Kann ein Objekt interne Daten „verstecken“? (Datenkapselung)

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

80

## Objekte anlegen, Attribute und Methoden



University of Applied Sciences

**Merke:** Funktionen in JavaScript sind Objekte, genau wie Arrays.

```
function func(a, b, c) {  
    // Implementierung  
}  
↔  
var func = function (a, b, c) {  
    // Implementierung  
}
```

Mittels des new Operators lassen sich Funktionsobjekte erzeugen. Diese beinhalten u.a. den internen Zustand einer Funktion. Die beiden unten stehenden Ausdrücke tun beide dasselbe. Es wird eine Funktion aufgerufen und das Ergebnis gespeichert. Nur im linken Fall wird das zugehörige Funktionsobjekt explizit erzeugt, im rechten (gebräuchlichen) Fall wird darauf verzichtet (weil meist das Funktionsobjekt an sich nie in der imperativen Programmierung benötigt wird).

```
var f = new func();  
var erg = f.call(1, 2, 3);
```

```
var erg = func(1, 2, 3);
```

Dadurch das Funktionen in JavaScript Objekte wie alle anderen sind und über ein sogenanntes Prototyping in eine hierarchische Vererbungsbeziehung gesetzt werden können, lassen sich aller vier aufgestellten Kontrollfragen zur Objektorientierung positiv beantworten. Dies soll nachfolgend gezeigt werden:

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

81

## Objekte anlegen, Attribute und Methoden



University of Applied Sciences

```
function Auto(v, g, s) { // Konstruktor  
    // Attribute (Datenfelder)  
    this.v = v;  
    this.gear = g;  
    this.steer = s;  
  
    // Methoden der Auto-"Klasse"  
    Auto.prototype.gearShift = function(d) { this.gear = d };  
    Auto.prototype.accelerate = function(d) { this.v += d };  
    Auto.prototype.decelerate = function(d) { this.v -= d };  
    Auto.prototype.head = function(d) { this.steer += d };  
    Auto.prototype.toString = function()  
        return "Geschwindigkeit: " + this.v + " km/h; " + this.gear + ". Gang; "  
        + "Lenkradeinschlag: " + this.steer + "°; "  
    }  
  
    // Neues Auto mit 50 km/h, im 3. Gang, kein Lenkradeinschlag (0°)  
    var mycar = new Auto(50, 3, 0);
```

Aus jeder Funktion kann also mittels des new Operators ein Objekt angelegt werden. Diesem Objekt können Attribute mittels der this Referenz zugewiesen werden. Über die prototype Referenz können Methoden an ein Objekt zugewiesen werden.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

82

## Objekte voneinander ableiten



University of Applied Sciences

```
// Ableiten einer Cabrio-“Klasse” aus der Auto-“Klasse”
Cabrio.prototype = new Auto();
Cabrio.prototype.constructor = Cabrio;

// Konstruktor
function Cabrio(v, g, s, o) {
    Auto.prototype.constructor.call(this, v, g, s);

    // public Attribute (Datenfelder) der Auto-“Klasse”
    this.open = o;
}

// Methoden der Cabrio-“Klasse”
Cabrio.prototype.openRoof = function() { this.open = true }
Cabrio.prototype.closeRoof = function() { this.open = false }
Cabrio.prototype.toString = function() {
    return Auto.prototype.toString.call(this) +
        (this.open ? "Dach offen" : "Dach geschlossen") + "; ";
}

// Ein Cabrio anlegen: v = 50 km/h, 3. Gang, 0° Lenkradeinschlag, Dach offen
var myCar = new Cabrio(50, 3, 0, true);
```

Die Vererbung in JavaScript erfolgt also ebenfalls mittels der prototype Referenz.  
Voneinander abgeleitete „Klassen“ sind also über prototype Referenzen miteinander verknüpft.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

83

## Kann ein Objekt Daten verstecken? Nicht von außen einsehbarer interner Zustand



University of Applied Sciences

```
// Ableiten einer PIN geschützten Auto-“Klasse” namens KeyCar
KeyCar.prototype = new Auto();
KeyCar.prototype.constructor = KeyCar;

// Konstruktor
function KeyCar(v, g, s, p) {
    Auto.prototype.constructor.call(this, v, g, s);

    // privates Attribut (Datenfeld)
    var pin = p;

    // privilegierte Methode
    this.toString = function() {
        return Auto.prototype.toString.call(this) + "PIN: " + pin + "; ";
    }
}

// Anlegen eines KeyCar Objekts
var keycar = new KeyCar(50, 3, 0, 123);
document.writeln("KeyCar erzeugt: " + keycar + "<br>");
document.writeln("PIN des KeyCars von außen abfragen ergibt: " + keycar.pin);
```

Privilegierte Methoden können auf private Variablen und Methoden zugreifen und sind selber für public-Methoden und außerhalb des Objekts zugreifbar. Privilegierte Methoden können gelöscht oder ersetzt werden, aber sie können nicht geändert werden.

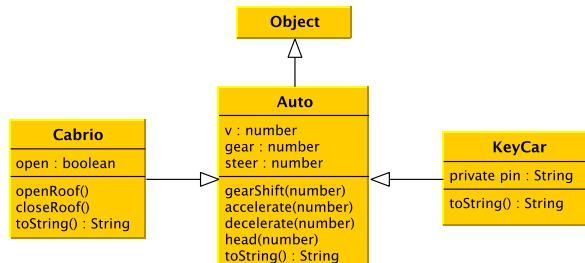
Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

84

## Kann ein Objekt mehrere Typen haben? Also polymorph sein ...

```
// Polymorphe Check
var mycab = new Cabrio(50, 3, 0, true);

document.writeln("mycab Instanz von Cabrio: " + (mycab instanceof Cabrio));
document.writeln("mycab Instanz von Auto: " + (mycab instanceof Auto));
document.writeln("mycab Instanz von KeyCar: " + (mycab instanceof KeyCar));
```



Ja!!! Polymorphie ist möglich in JS. mycab ist Instanz von Auto und Object aber nicht von KeyCar.

Das ist das typische OO Verständnis von Objekt-Polymorphie.

## JavaScript ist also eine objektorientierte Sprache!

- Es können Objekte angelegt und diesen Attribute und Methoden zugewiesen werden!
- Es können Objekte voneinander abgeleitet werden. Das Sohnobjekt kann dabei auf die Attribute und Methoden des Vaters zugreifen? (Vererbung)
- Ein Objekt kann gleichzeitig Instanz mehrerer Typen sein. Ein Sohnobjekt ist immer auch vom Typ des Vaters? (Polymorphie)
- Ein Objekt kann private Daten deklarieren! (Datenkapselung)

Diese Form der prototypenbasierten Objektorientierung ist sicher gewöhnungsbedürftig, aber entspricht vollkommen den objektorientierten Prinzipien (Datenkapselung, Vererbung und Objekt-Polymorphie).



Das gesamte Beispiel kann unter folgendem Link noch einmal nachvollzogen werden.

<http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/OOFeatures/oo.html>

## Programmiersprache JavaScript



University of Applied Sciences



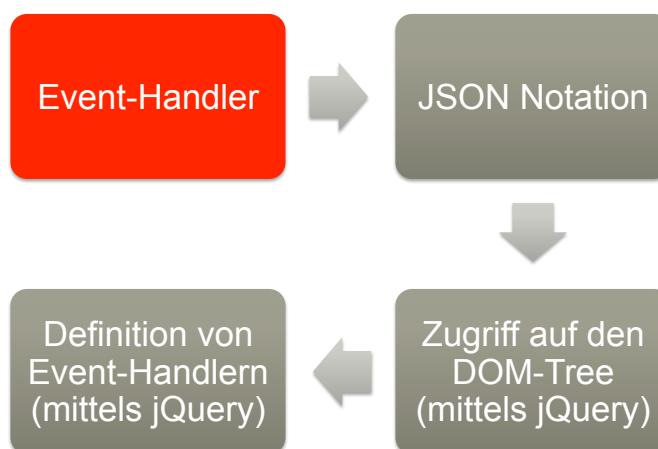
Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

87

## Sprachbesonderheiten



University of Applied Sciences



Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

88

## Event-Handler (inkl. Auswahl einiger Handler)



University of Applied Sciences

Event-Handler 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. Dort wird auch festgelegt, in welchen HTML-Tags welcher Event-Handler vorkommen darf.

Event	Beschreibung	Tag
onblur	Ein zuvor aktiviertes Element wird verlassen.	<a> <area> <button> <input> <label> <select> <textarea>
onchange	Ein Element hat einen geänderten Wert erhalten.	<input> <select> <textarea>
onclick	Beim Anklicken eines Elements	In diversen Tags
onload	Beim Laden einer Datei	<iframe> <body>
onmouse{down, move, out, over}	Korrespondierende Mausereignisse (drücken, (heraus)bewegen, verharren)	In diversen Tags
onsubmit	Beim Absenden eines Formulars	<form>
javascript	Dies ist kein Event-Handler im engeren Sinn. Es handelt sich um eine Syntax, die eingeführt wurde, um JavaScript-Code als Verweisziel zu notieren.	<a> <area>

Prof. Dr. rer. nat. Nane Kratzke

Praktische Informatik und betriebliche Informationssysteme

89

## Event Handler Demonstration



University of Applied Sciences

**Datei:** handler.html

```
[...]
<script type="text/javascript" src="handler.js"></script>
[...]

<h1 onmouseover="mouseover()">Fahre über mich</h1>
<p id="miracle"></p>
```

**Datei:** handler.js

```
var klick = 0; var over = 0;

function click() {
    var text = document.createTextNode("Ihr " + ++klick + "ter Mouse over! ");
    document.getElementById("miracle").appendChild(text);
    br = document.createElement("br");
    document.getElementById("miracle").appendChild(br);
}
```



Auf diese Art und Weise können Sie mittels clientseitig ausgeführten Skripten eine Nutzerinteraktion erzielen, wie dieses Beispiel zeigt.

<http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/EventHandler/handler.html>

Prof. Dr. rer. nat. Nane Kratzke

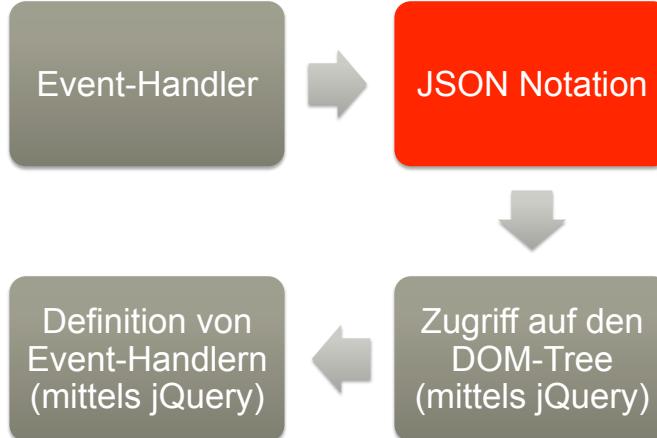
Praktische Informatik und betriebliche Informationssysteme

90

## Sprachbesonderheiten



University of Applied Sciences



Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme | 91

## JSON Notation



University of Applied Sciences

Die **JavaScript Object Notation**, kurz JSON, ist ein kompaktes Datenformat in für Mensch und Maschine einfach lesbarer Textform zum Zweck des Datenaustauschs zwischen Anwendungen. Jedes gültige JSON-Dokument soll ein gültiges JavaScript sein und per eval() interpretiert werden können.

Davon abgesehen ist JSON aber unabhängig von der Programmiersprache. Parser existieren in praktisch allen verbreiteten Sprachen.

Sie haben schon gesehen, dass die beiden folgenden Zeilen, dasselbe Array erzeugen. Die zweite Zeile ist einfach nur eine Kurznotation für die erste (und nebenbei JSON :-).

```
var a = new Array("Hello", " ", "World");  
var a = ["Hello", " ", "World"];
```

Eine vergleichbare Notation gibt es in JavaScript auch, um Objekte anlegen zu können. Die Kurznotation für Arrays und Objekte macht den Kern der JSON Notation aus.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme | 92

## JSON Notation

### Beispiel: Objekte anlegen



University of Applied Sciences

Ein Objekt dieser Art, kann auch

```
var p = new Object();
p.vorname = "Maren";
p.nachname = "Musterfrau";
p.hallo = function() {
    return "Hallo mein Name ist " + this.nachname
}
```

als JSON wie folgt ausgedrückt werden

```
var p = {
    "vorname" : "Maren",
    "nachname" : "Musterfrau",
    "hallo" : function() {
        return "Hallo mein Name ist " + this.nachname
    }
}
```

**Hinweis:** Mit JSON können also nicht nur Daten, sondern auch JavaScript Code ausgedrückt (und serialisiert) werden.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

93

## JSON Notation in a Nutshell



University of Applied Sciences

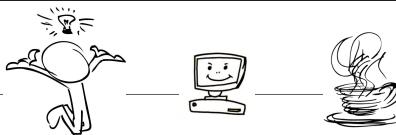
In Anlehnung an Wikipedia:  
[http://de.wikipedia.org/wiki/JavaScript\\_Object\\_Notation](http://de.wikipedia.org/wiki/JavaScript_Object_Notation)

Datentyp	Literal	Erläuterung
Nullwert	null	Wird durch das Schlüsselwort null dargestellt.
Boolescher Wert	true, false	Wird durch die Schlüsselwörter true und false dargestellt. Dies sind keine Zeichenketten. Sie werden daher, wie null, nicht in Anführungszeichen gesetzt.
Zahl	0–9	Ist eine Folge der Ziffern 0–9. Diese Folge kann durch ein negatives Vorzeichen - eingeleitet und einen Dezimalpunkt . unterbrochen sein. Die Zahl kann durch die Angabe eines Exponenten e oder E ergänzt werden, dem ein Vorzeichen + oder - und eine Folge der Ziffern 0–9 folgt.
Zeichenkette	“ ”	Beginnt und endet mit doppelten geraden Anführungszeichen (“”). Sie kann Unicode-Zeichen und Escape-Sequenzen enthalten.
Array	[ ]	Beginnt mit [ und endet mit ]. Es enthält eine durch Komma getrennte, geordnete Liste von Werten, gleichen oder verschiedenen Typs. Leere Arrays sind zulässig.
Objekt	{ }	Beginnt mit { und endet mit }. Es enthält eine durch Komma getrennte, ungeordnete Liste von Eigenschaften. Objekte ohne Eigenschaften ("leere Objekte") sind zulässig.
Eigenschaft	key:val	Besteht aus einem Schlüssel und einem Wert, getrennt durch einen Doppelpunkt (key:val). Die Schlüssel aller Eigenschaften in einem Objekt müssen eindeutig, also paarweise verschieden sein. <ul style="list-style-type: none"> <li>• Der Schlüssel (key) ist eine Zeichenkette.</li> <li>• Der Wert (val) ist ein Objekt (und kann damit auch eine Funktion sein), ein Array, eine Zeichenkette, eine Zahl oder einer der Literale true, false oder null.</li> </ul>

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

94

## Miniübung:



Gegeben ist folgendes JavaScript. Drücken Sie die Datenstruktur persons in JSON aus.

```
var p1 = new Object(); p1.vn = "Max"; p1.nn = "Mustermann";
var p2 = new Object(); p2.vn = "Maren"; p2.nn = "Musterfrau";
var p3 = new Object(); p3.vn = "Tessa"; p3.nn = "Loniki";

var persons = new Array(p1, p2, p3);
```

Lösung:



Sie können unter diesem Link, das Beispiel auch noch einmal mit Methoden nachvollziehen.

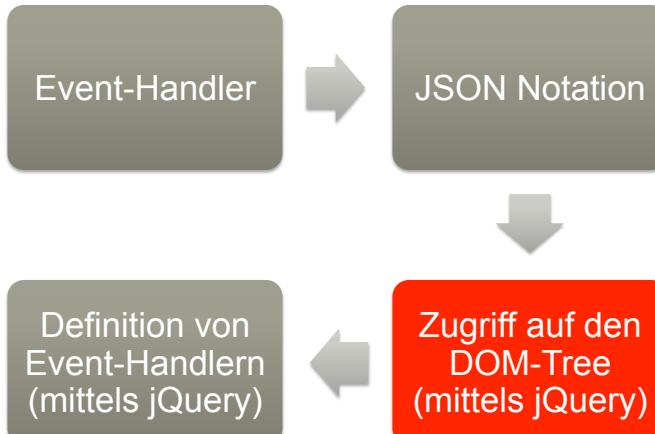
<http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/JSON/json.html>

Prof. Dr. rer. nat. Nane Kratzke

Praktische Informatik und betriebliche Informationssysteme

95

## Sprachbesonderheiten



Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

96

## jQuery



University of Applied Sciences

jQuery ist eine freie, umfangreiche JavaScript-Klassenbibliothek, die komfortable Funktionen zur DOM-Manipulation und -Navigation zur Verfügung stellt.

jQuery ist einer der meistverwendeten JavaScript-Bibliotheken und damit ein Quasi-Standard in der clientseitigen Programmierung.

jQuery beinhaltet dabei u.a. nachfolgende Funktionalitäten:

- Elementselektion im Document Object Model (vergleichbar CSS3 Selektoren, siehe dort) inkl. mächtiger Manipulationsmöglichkeiten des DOM-Trees
- Animations-Effekte
- Komfortables Event-System
- Ajax-Funktionalitäten



Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

97

## jQuery – Hello World



University of Applied Sciences

Um mit jQuery Arbeiten zu können, muss auf jeder HTML Seite die jQuery Bibliothek importiert werden. Anschließend kann man die jQuery Funktionen zum

- **Elementzugriff (DOM-Tree)**
- **CSS** sowie zur
- **Ereignisbehandlung** (inkl. Ajax)

nutzen. Zentrale Funktion ist dabei die `$( )` Funktion in jQuery (wie mittlerweile in diversen anderen JavaScript Frameworks auch).

jQuery Bibliothek importieren

```
<script type="text/javascript" src="js/jquery.min.js"></script>
<script type="text/javascript">
$(document).ready(function() {
    $("#msgid").hide().html("This is Hello World by JQuery");
    $("#msgid").css("color", "red");
    $("#msgid").fadeIn(1000).fadeOut(1000).fadeIn(2000);
});
</script>
```

Event-Handler registrieren

Elemente ändern

CSS setzen und abfragen

Effekte auf Elementen auslösen

Elemente im DOM-Tree selektieren

[http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/jQuery/hello\\_world.jquery.html](http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/jQuery/hello_world.jquery.html)

Prof. Dr. rer. nat. Nane Kratzke

98

Praktische Informatik und betriebliche Informationssysteme

# Handout zur Vorlesung DBWP – Unit JavaScript

## jQuery – Elemente manipulieren (descriptive)



University of Applied Sciences

### DOM Insertion Around

- `.wrap()` – Klammt ein Element mit einem Tag
- `.wrapInner()` – Klammt den Inhalt eines Elementes mit einem Tag
- `.wrapAll()` – Klammt mehrere Elemente mit einem Tag
- `.unwrap()` – Entfernt ein umschließendes Tag eines Elements
- ...

Erarbeiten Sie sich bitte  
diese Funktionen anhand  
der API.  
Ggf. relevant für Klausur!

### DOM Insertion Inside

- `.html()` – Setzt den Inhalt (oder fragt diesen ab) innerhalb eines Elements
- `.append()` – Hängt Inhalt innerhalb eines Elements an
- `.prepend()` – Fügt Inhalt am Anfang eines Elements ein
- `.text()` – Liest und setzt den Inhalt innerhalb eines Elements inklusive aller Subelemente
- ...

### DOM Insertion Outside

- `.after()` – Fügt ein Element hinter einem anderen Element ein
- `.before()` – Fügt ein Element vor einem anderen Element ein
- ...

### DOM Removal

- `.empty()` – Löscht den Inhalt eines Elements (Tags)
- `.detach()` – Entfernt ein komplettes Element (Tag) samt Inhalt und Subtree
- `.remove()` – Löscht ein komplettes Element (Tag) samt Inhalt und Subtree

In Anlehnung an jQuery API:  
<http://api.jquery.com/category/manipulation/>

Prof. Dr. rer. nat. Nane Kratzke

Praktische Informatik und betriebliche Informationssysteme

99

## Beispiel: DOM Tree Manipulation



University of Applied Sciences

Gegeben ist der linke HTML Code. Dieser soll mittels jQuery in den rechten gewandelt werden.

```
<div id="ps">
<p class="li">A</p>
<p class="li">B</p>
<p class="li">C</p>
<p class="li">D</p>
<p class="li">E</p>
</div>
```

```
$( "#ps p.li" ).wrapInner(" <li></li> " );
```

```
<div id="ps">
<p class="li"><li>A</li></p>
<p class="li"><li>B</li></p>
<p class="li"><li>C</li></p>
<p class="li"><li>D</li></p>
<p class="li"><li>E</li></p>
</div>
```



[http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/jQuery/dom\\_transform.html](http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/jQuery/dom_transform.html)

```
<div id="ps">
<ul>
<li>A</li>
<li>B</li>
<li>C</li>
<li>D</li>
<li>E</li>
</ul>
</div>
```

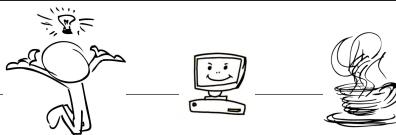
```
$( "#ps li" ).wrapAll(" <ul></ul> " );
```

```
<div id="ps">
<li>A</li>
<li>B</li>
<li>C</li>
<li>D</li>
<li>E</li>
</div>
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

100

## Miniübung:



University of Applied Sciences

Gegeben ist folgender HTML-Code. Geben Sie bitte eine jQuery Manipulation an, die rechts stehendes Ergebnis produziert:

```
<table border=1>
<tr><th>Vorname</th><th>Nachname</th></tr>
<tr><td>Max</td><td>Mustermann</td></tr>
<tr><td>Maren</td><td>Musterfrau</td></tr>
<tr><td>Tessa</td><td>Loniki</td></tr>
<tr><td>Kon</td><td>Tiki</td></tr>
</table>
```

```
<ul>
<li>Max Mustermann</li>
<li>Maren Musterfrau</li>
<li>Tessa Loniki</li>
<li>Kon Tiki</li>
</ul>
```



Sie können unter diesem Link, das Beispiel im Detail nachvollziehen.

[http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/jQuery/table\\_transform.html](http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/jQuery/table_transform.html)

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

101

## jQuery – Stylesheets manipulieren (presentational)



University of Applied Sciences

Ähnlich wie mit den Methoden zum Manipulieren des DOM-Trees lassen sich auch die Stylesheet Informationen mittels jQuery clientseitig manipulieren. jQuery sieht hierzu die folgenden drei Methoden vor:

```
.css( propertyName )
```

Liefert den Wert eines style property für das erste Element in der Treffermenge des Selektors.

```
.css( propertyName, value )
.css( map )
```

Setzt den Wert eines style property für alle Elemente in der Treffermenge des Selektors. Soll mehr als ein style property gesetzt werden, kann mit einem Mapping map gearbeitet werden.

```
.css( propertyName, function ( i, val ) )
```

Setzt den Wert eines style property für alle Elemente in der Treffermenge des Selektors mittels einer Funktion. i ist dabei der Index des Elements und val der aktuell gesetzte Wert des Elements für das style property.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

102

## Beispiel: CSS Manipulation



University of Applied Sciences

Gegeben ist der nachfolgende HTML und CSS Code. Mittels jQuery soll sichergestellt werden, dass alle roten Texte blau werden. Wie stellen Sie das an?

HTML:

```
<ul>
<li class="a">A</p>
<li class="b">B</p>
<li class="c">C</p>
<li class="a">D</p>
<li class="b">E</p>
</ul>
```

CSS:

```
.a { color : red }
.b { color : green }
.c { color : blue }
```



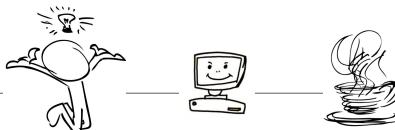
Sie können unter diesem Link, das Beispiel im Detail nachvollziehen.

[http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/jQuery/css\\_transform.html](http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/jQuery/css_transform.html)

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

103

## Miniübung:



University of Applied Sciences

Gegeben ist folgender CSS-Code. Realisieren Sie diesen bitte mittels einer jQuery Emulation:

```
h1, h2, h3.important {
    font-style: italic;
    font-weight: bold;
    color: blue;
}

p.example {
    color: darkgrey;
}

#content {
    background-color: lightgray;
}
```

CSS



[http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/jQuery/css\\_reference.html](http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/jQuery/css_reference.html)

jQuery



[http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/jQuery/css\\_jquery\\_emulation.html](http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/jQuery/css_jquery_emulation.html)

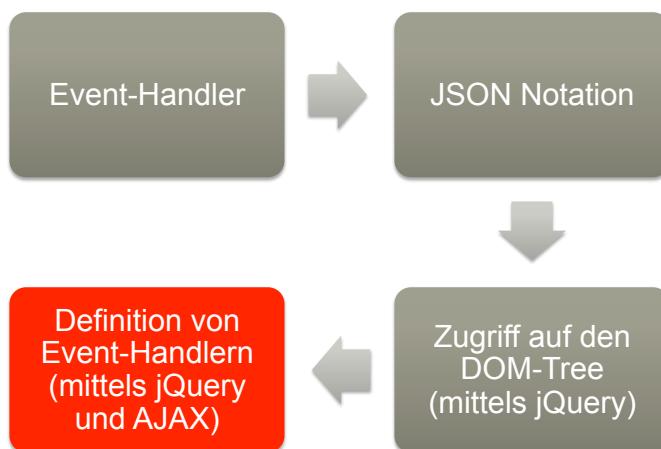
Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

104

## Sprachbesonderheiten



University of Applied Sciences



Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme | 105

## jQuery Events



University of Applied Sciences

Alle in HTML definierten Handler, haben auch jQuery Pendants. Diese können dazu genutzt werden, mittels jQuery Callback Funktionen für Browserereignisse zu registrieren.

Browser Events	Document Loading Events	Form Events
<ul style="list-style-type: none"><li>• .error()</li><li>• .resize()</li><li>• .scroll()</li></ul>	<ul style="list-style-type: none"><li>• .load()</li><li>• .ready()</li></ul>	<ul style="list-style-type: none"><li>• .change()</li><li>• .focus()</li><li>• .submit()</li><li>• ...</li></ul>
Keyboard Events		Mouse Events
<ul style="list-style-type: none"><li>• .focusin()</li><li>• .focusout()</li><li>• .keydown()</li><li>• .keypress()</li><li>• .keyup()</li><li>• ...</li></ul>		<ul style="list-style-type: none"><li>• .click()</li><li>• .dblclick()</li><li>• .hover()</li><li>• .mouse{enter/leave}()</li><li>• .mouse{down/up}()</li><li>• ...</li></ul>

Details unter folgendem Link:  
<http://api.jquery.com/category/events/>

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme | 106

## Beispiel: Ein Keyboard Sniffer



University of Applied Sciences

Am Beispiel eines Keyboard Sniffers soll das Event Handling mittels jQuery verdeutlicht werden. Hätten Sie gedacht, dass Sie mittels JavaScript in drei Zeilen einen Keyboard Sniffer schreiben können, der dazu geeignet ist, z.B. Passworteingaben mitzuschneiden?

**Hinweis:** Das vorliegende Beispiel ist dazu übrigens nicht in der Lage – aber demonstriert relevante Aspekte! Es dient neben der Demonstration des Event Systems auch dem Schärfen ihres Security Empfindens!

```
<script type="text/javascript" src="js/jquery.min.js">
<script type="text/javascript">
$(document).ready(function() {
    $(window).keydown(function (event) {
        $("#keydowns").text(function (i, old) {
            return old + String.fromCharCode(event.keyCode);
        });
    });
</script>

<h1>Keyboard Sniffer</h1>
<div id="keydowns" />
```

Am Window Objekt wird das keydown Event mit einer anonymen Callback Funktion verknüpft. Jede Tasteneingabe im Browserfenster wird jetzt an diese anonyme Funktion weitergeleitet (rot).

In der Callback Funktion wird der Textinhalt des DOM Elements mit der ID keydown ausgelesen und mit String Entsprechung des eingegebenen KeyCodes verknüpft. Anschließend wird dieser so um die letzte Eingabe verlängerte Text wieder ausgegeben (blau).



Sie können unter diesem Link, das Beispiel im Detail nachvollziehen.

<http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/JQuery/keysniffer.html>

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

107

## jQuery und Ajax

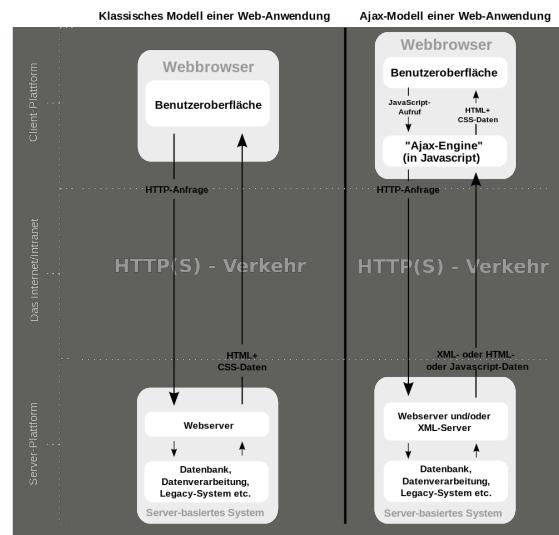


University of Applied Sciences

Ajax steht für die Wortfolge „Asynchronous JavaScript and XML“.

Es bezeichnet ein Konzept der **asynchronen** Datenübertragung zwischen einem Browser und dem Server.

Dieses ermöglicht es, HTTP-Anfragen durchzuführen, während eine HTML-Seite angezeigt wird, und die Seite zu verändern, ohne sie komplett neu zu laden. Viele Anwendungen von Ajax werden dazu eingesetzt, im Webbrowser ein desktopähnliches Verhalten zu simulieren, wie beispielsweise Popup-Fenster.



Quelle: [http://de.wikipedia.org/wiki/Ajax\\_\(Programmierung\)](http://de.wikipedia.org/wiki/Ajax_(Programmierung))

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

108

## jQuerys (clientseitige) AJAX Unterstützung



University of Applied Sciences

jQuery bietet eine komfortable AJAX Unterstützung mittels der nachfolgenden Funktionen an, die es einem ersparen sich mit den Details des zu Grunde liegenden XMLHttpRequest (XHR) Protokolls im Einzelnen befassen zu müssen.



Methode	Beschreibung
<code>jquery.get()</code>	Lädt Daten von einem Server mittels eines HTTP GET Requests.
<code>jquery.post()</code>	Lädt Daten von einem Server mittels eines HTTP POST Requests.
<code>jquery.getJSON()</code>	Lädt JSON codierte Daten von einem Server mittels eines HTTP GET Requests.
<code>jquery.getScript()</code>	Lädt ein Skript von einem Server mittels eines HTTP GET Requests und führt dieses anschließend aus.
<code>.load()</code>	Lädt Daten (HTML) von einem Server und fügt diese in das selektierte Element ein.

Quelle: <http://api.jquery.com/category/ajax/shorthand-methods/>

Prof. Dr. rer. nat. Nane Kratzke

109

Praktische Informatik und betriebliche Informationssysteme

## Funktionsweise der AJAX Methoden



University of Applied Sciences

Jede der genannten Methoden hat dabei im Kern die folgende Signatur, die am Beispiel der `getJSON()` Methode exemplarisch veranschaulicht werden soll:

```
jquery.getJSON(url [, data] [, callback(data, status, xhr)])
```

- **url** : Eine URL (String) an die der Request gesendet wird
- **data** : Eine map oder ein String, die mit dem Request versendet werden (z.B. get Parameter)
- **callback** : Eine (zumeist anonyme) Callback Funktion, die aufgerufen wird, wenn der Request erfolgreich bearbeitet wurde.
  - **data** : die Daten als Antwort auf den Request kommen (im Falle von getJSON ein Object oder Array in JSON Notation).
  - **status** : der HTTP Status der Anfrage (üblicherweise bei Erfolg 200 OK)
  - **xhr** : Ein XMLHttpRequest Objekt, dass Details zur Kommunikation offenbart.

*Hinweis:* Aus Sicherheitsgründen verfolgen Browser im Allgemeinen eine sogenannte **Same Origin Policy**. Diese besagt, dass das Nachladen von Daten nur von der Domain gestattet ist, von der auch das Dokument ursprünglich geladen wurde. So wird versucht zu vermeiden, dass nicht vertrauenswürdige Daten von unkontrollierbaren Quellen nachgeladen werden können. Mittels Cross-Domänen Ansätzen, wie beispielsweise JSONP kann dies aber umgangen werden. Hierauf wird in dieser Vorlesung aber nicht eingegangen.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

110

## Beispiel: AJAX basierter Keyboardsniffer



University of Applied Sciences

Das Prinzip von AJAX soll anhand einer kleinen Erweiterung unseres Keyboard Sniffer Beispiels erfolgen. Nun sollen in diesem Beispiel nicht nur die Tastatureingaben eingelesen, sondern auch an einen Server weitergereicht werden. Der Server soll die Anfrage mit einer JSON Datei beantworten, die bereits mehrere eingelesene Passwörter beinhaltet und diese sollen in das HTML Dokument eingebettet werden.

```
<script type="text/javascript">
$(document).ready(function() {
    $(window).keydown(function (event) {
        $("#keydowns").text(function (i, old) {
            var text = old + String.fromCharCode(event.keyCode);
            if (text.indexOf("AJAX") != -1) {
                var send = { "keycodes" : text };
                $.getJSON('ajax/test.json', send, function(data) {
                    var pwds = "";
                    $.each(data, function(i, p) {
                        pwds += "<li>" + p.name + " (has password: " + p.password + "<br>"); 
                    });
                    $("#JSON").html("Das habe ich gesendet: " + send + "<br>"); 
                    $("#JSON").append("Das habe ich erhalten: <ul>" + pwds + "</ul>");
                });
            [...]
        });
    });
    </script>
<h1>Keyboard Sniffer</h1>
<div id="keydowns" /> <div id="JSON" />
```

Mittels der `getJSON()` Methode wird eine JSON Datei geladen (grün), sollte der Nutzer einmal „AJAX“ eingegeben haben.

Dabei werden alle bisherigen Tastatureingaben an den Server transferiert (rot).

Die Callback Routine erhält die JSON Daten im Parameter `data` und wertet diese aus (blau).



Sie können unter diesem Link, das Beispiel im Detail nachvollziehen.

<http://webtech.fhl.de.s3-website-us-east-1.amazonaws.com/JavaScript/jQuery/ajaxkeysniffer.html>

Prof. Dr. rer. nat. Nane Kratzke

111

Praktische Informatik und betriebliche Informationssysteme

## Zusammenfassung



University of Applied Sciences

- **JavaScript**
  - Einordnung von JavaScript (Programmier- und Laufzeitparadigma)
  - Einbinden der clientseitigen Sprache JavaScript
  - Dynamische Typisierung und Operatoren
  - Ablaufsteuerungen, Funktionen und funktionales Programmieren (Lambda Funktionen, höherwertige Funktionen)
  - Datenstrukturen (Listen und Objekte/Maps)
  - Prototypenbasierte Objektorientierung
- **Sprachbesonderheiten**
  - Eventhandler
  - JSON Notation
  - DOM-Tree Manipulation mit jQuery
  - Event-Handling und AJAX mit jQuery



Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

112