Doing Web Apps

# JavaScript 2

Autor: Rüdiger Marwein
Letzte Änderung: 2014-10-24
Version: 1.1

Dieses Dokument darf – mit Nennung des Autoren - frei vervielfältigt, verändert und weitergegeben werden.

Der Inhalt ist sorgfältig recherchiert, mit dem Dokument ist jedoch keinerlei Garantie auf Fehlerfreiheit gewährleistet.

Dieser Inhalt ist unter einem Creative Commons Namensnennung Lizenzvertrag lizenziert. Um die Lizenz anzusehen, gehen Sie bitte zu http://creativecommons.org/licenses/by/2.0/de/ oder schicken Sie einen Brief an Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

# Inhaltsverzeichnis

1.Arrays / Listen....3
2.Objekte....3
3.Array und Objekte kombinieren....3
4.Schleifen....4
5.Die While-Schleife....4
6.Die For-Schleife....4
7.break und continue....5
8.Dokument-Eigenschaften....5
9.Fenster-Eigenschaften....6
10.DOM Element-Eigenschaften....7
11.Nachdem der DOM fertig aufgebaut ist....8
12.Stylesheet-Einstellungen ändern....8
13.Eine CSS-Klasse hinzufügen....9

# 1.Arrays / Listen

Arrays sind Listen von Werten. Die Werte können beliebigen Typs sein; auch innerhalb einer Liste.

Array deklarieren:

```
var leer  = []; // leeres Array
var liste = [3]; // Array mit einem Wert „3"
var liste = ["Hans", "Peter"]; // Array mit 2 Werten
// Objekt-Notation:
var liste = new Array(); // leeres Array
var liste = new Array("Hans"); // Array mit einem Wert „Hans"
var liste = new Array(3); // Array mit 3 leeren Werten(!!)
var liste = new Array(1, 2, 3); // Array mit den Werten 1, 2 und 3
```

Beachten Sie, dass jeweils hinter dem letzten Element kein Komma stehen darf.

Einem Array weitere Elemente hinzufügen

```
liste.push("Gaby");
```

Anzahl der Elemente eines Arrays ermitteln

```
liste.length;
```

Übrigens: Zeichenketten sind Reihungen von Buchstaben und haben ebenfalls die Eigenschaft „length".

Auf Elemente eines Arrays greift man mit eckigen Klammern zu. Array-Einträge werden von 0 (null) aufsteigend nummeriert.

Erstes Element: `liste[0]`

Zweites Element: `liste[1]`

# 2.Objekte

Anonyme Objekte

```
var leer = {};
var benutzer = {"vorname": "Hans", "nachname": "Müller"};
```

Beachten Sie, dass jeweils hinter dem letzten Element kein Komma stehen darf.

Objekte in dieser Notation werden gern verwendet um mehrere Konfiguration-Eigenschaften auf einmal zu setzen.

Auf die Eigenschaften eines Objektes greifen Sie mit Punkt zu.

Der Vorname: `benutzer.vorname`

# 3.Array und Objekte kombinieren

```
var benutzer = {
  vorname: "",
  nachname: "",
  interessen: [
    "Sport", "Lesen", "Fernsehen"
  ]
  rechte: [
```

```
    {title: "read", description: "Lese-Berechtigung"},
    {title: "write", description: "Schreib-Berechtigung"}
  ]
}
```

Beachten Sie, dass jeweils hinter dem letzten Element kein Komma stehen darf.

Die Schreibweise für den Zugriff ist somit kombiniert.

Benutzer → Recht 1 → Beschreibung

```
benutzer.rechte[0].description
```

# 4.Schleifen

Schleifen wiederholen einen Programm-Abschnitt so lange, bis eine Bedingung nicht mehr zutrifft.

Sie sind nützlich um mathematische Formeln nachzustellen, Listen abzuarbeiten, Schrittweise Prüfungen zu machen und vieles mehr.

Es gibt 2 hauptsächlich eingesetzte Schleifen-Arten: while und for.

# 5.Die While-Schleife

```
var a = 1, b = 1;
while (a < 4) {
  ++a;
  b *= a;
}
alert(b);
```

In Worten: „Solange der Zähler „a“ kleiner als 4 ist, erhöhe den Zähler um eins und multipliziere b mit dem Wert des Zählers“

Würde man die 3 Schleifendurchgänge „auffalten“, würde dies so aussehen:

```
var a = 1, b = 1; // a=1, b=1
  a++;      // a=2, b=1
  b *= a;   // a=2, b=2
// Schleife beginnt von vorn, denn a < 4
  a++;      // a=3, b=2
  b *= a;   // a=3, b=6
// Schleife beginnt von vorn, denn a < 4
  a++;      // a=4, b=6
  b *= a;   // a=4, b=24
// Schleife endet, da a >= 4
alert(b);         // 24
```

# 6.Die For-Schleife

For-Schleifen lassen sich am besten einsetzen, wenn es sich um Zähl-Schleifen handelt.

Im Schleifen-Kopf werden direkt Start-Wert, Fortsetzungsbedingung und Zähler notiert.

```
for ( var a=1; a<4; a++ ) {
  b *= a;
}
```

In Worten: „Beginne mit dem Zähler a=1 und solange a < 4 ist, erhöhe a um 1. Multipliziere in jedem Schritt b mit dem Wert des Zählers“

Um Arrays und Objekte zu durchlaufen, existiert eine besondere Notation dieser Schleife.

```
var fahrzeuge = [
  {typ: "Auto", marke:"Audi"},
  {typ: "Motorrad", marke:"Yamaha"},
  {typ: "Auto", marke:"Toyota"},
  {typ: "Fahrrad", marke:"Shimano"}
];
var autos = 0;
for ( var key in fahrzeuge ) {
  var fahrzeug = fahrzeuge[key];
  if(fahrzeug.typ === "Auto") {
      autos++;
  }
}
alert("Autos in der Liste: " + autos);
```

# 7.break und continue

Es ist möglich die Abarbeitung einer Schleife zu beeinflussen, indem man sie vorzeitig beendet (`break`) oder zum nächsten Schleifendurchlauf fortschreitet (`continue`).

```
var ignore = 5,
    evilValue = 13;
    a = 0;
while(a < 100) {
  a++;
  if(a <= ignore) {
    continue;
  }

  a += 3; // wird erst erreicht, wenn a > ignore
  if(a == evilValue) {
    break;
  }
}
```

Die o.g. Schleife erreicht nicht die Ende-Bedingung a >= 100. Der Endwert von a ist 13.

# 8.Dokument-Eigenschaften

Das Objekt „document“ ist stets verfügbar und repräsentiert den aktuellen DOM Baum mit dessen Elementen.

Das document-Objekt hat eine Reihe von Eigenschaften und Funktionen.

**Die wichtigsten Eigenschaften sind:**

Der Dokument Titel bzw Titel des Browser-Fensters.

```
var documentTitel = document.title;
document.title = "Neuer Fenster-Titel";
```

Ladezustand der Seite (uninitialized / loading / interactive / complete):

```
var state = document.readyState;
```

**Die wichtigsten Funktionen sind:**

In das Dokument schreiben (während des Dokument-Aufbaus möglich)

```
document.write( "Text-Ausgabe" );
```

Ein HTML-Element erzeugen

```
var divElem = document.createElement( "div" );
```

Ein Element anhand seiner ID finden

```
var element = document.getElementById( "die-kennung" );
```

Alle Elemente eines Tag-Namens finden

```
document.getElementByTagName( "img" );
```

Finden eines Elements mit einer speziellen Klasse ( IE erst ab 9, sonst überall )

```
document.getElementsByClassName('der-typ');
```

# 9.Fenster-Eigenschaften

Das Objekt „window“ ist stets verfügbar und repräsentiert das Browser-Fenster.

Das window-Objekt hat eine Reihe von Eigenschaften und Funktionen.

**Die wichtigsten Eigenschaften sind:**

Die aktuelle URL

```
// http://www.heise.de/index.php?id=45#marke
var url = window.location.href;
var protocol = window.location.protocol; // http:
var host     = window.location.host;     // www.heise.de
var path     = window.location.pathname; // /index.php
var query    = window.location.search;   // ?id=45
var hash     = window.location.hash;     // #marke
```

**Die wichtigsten Funktionen sind:**

Eine Funktion nach einer gewissen Zeit (Millisekunden – 1000ms = 1s) ausführen

```
var timeout = window.setTimeout ( callbackFunktion, 1000 );
```

Eine zeitverzögerte Ausführung vorzeitig stoppen

```
window.clearTimeout(timeout);
```

Eine Funktion nach einer gewissen Zeit (Millisekunden) wiederholen

```
var interval = window.setInterval( callbackFunktion, 1000 );
window.clearInterval(timeout);
```

> Wichtig bei setInterval: Sollte der Code innerhalb der Callback-Funktion länger als die „Pause-Zeit“ dauern, wird trotzdem ein zusätzlicher Interval parallel gestartet. Die Funktion kann sich also „überholen“. Das Ergebnis kann dann nicht unbedingt vorhergesehen werden (race-condition).

Ein Fenster öffnen

```
// window.open( url, name, konfiguration);
var myWin = window.open( "popup.html", "Fenster",
```

```
"width=100,height=100,toolbar=yes,scrollbars=yes");
```

Ein benanntes Fenster in den Vordergund holen (u.U. Durch Browser-Einstellung verboten)

```
window.focus();
myWin.focus();
```

Das alert()- Popup

```
window.alert()
```

Das Dokument drucken (Druck-Dialog öffnen)

```
window.print()
```

Die zuvor aufgerufene Webseite öffnen

```
window.back()
```

# 10.DOM Element-Eigenschaften

Jeder DOM-Knoten hat bestimmte Eigenschaften. Hat man einen DOM-Knoten bspw via document.getElementById() geladen, kann man Element-Eigenschaften auslesen und sich von dort weiter durch den DOM-Baum bewegen.

```
var element = document.getElementById("die-kennung");
```

**Die wichtigsten Eigenschaften:**

Den HTML-Inhalt eines Knoten (ohne das umschließende Tag)

```
var inhalt = element.innerHTML;
element.innerHTML = "Neuer <strong>Inhalt</strong>";
```

Den Knoten in der Hierarchie über dem aktuellen Element.

```
element.parentNode
```

Das erste Element der Reihe von Elementen innerhalb des aktuellen (bspw das erste <li> Tag innerhalb eines <ul> Tags)

```
element.firstChild
```

Das letzte Element der Reihe von Elementen innerhalb des aktuellen (bspw das letzte <li> Tag innerhalb eines <ul> Tags)

```
element.lastChild
```

Das nächste Element neben dem aktuellen (bspw das nächste <li>, wenn Element ein <li> ist)

```
element.nextSibling
```

Das vorhergehende Element neben dem aktuellen (bspw das vorhergehenede <li>, wenn element ein <li> ist)

```
element.previousSibling
```

Alle Unter-Knoten (eine Ebene) eines Elements (bspw alle <li> eines <ul> Elements) als Array

```
element.childNodes
```

Ein Tag-Attribut auslesen

```
var element = document.getElementById('die-kennung');
var title = element.title;
var id = element.id;
...
```

**Die wichtigsten Funktionen:**

Ein Element an das Ende eines Tags anhängen (bspw ein <li> am Ende hinzufügen)

```
var div = document.createElement('div');
div.innerHTML = "Inhalt";
element.appendChild( div );
```

Ein Element aus einem anderen Element löschen

```
var first = element.firstChild;
element.removeChild( first );
```

Finde alle Tags eines bestimmten Typs innerhalb eines Elements

```
element.getElementsByTagName( "div" );
```

# 11.Nachdem der DOM fertig aufgebaut ist

Aktionen ausführen, nachdem das DOM Dokument fertig geladen ist.

Solange der DOM sich im Aufbau befindet, liefert die Eigenschaft document.readyState den Wert „incomplete“. Ist der DOM fertig aufgebaut, liefert sie „complete“. Somit lässt sich der Zeitpunkt mit eine „Timeout-Schleife“ bestimmen.

```
function whenReady(func){
  if(document.readyState !== "complete") {
    window.setTimeout('whenReady('+func+')',9);
  } else {
    func();
  }
}
whenReady(function() {
  alert('Dokument geladen');
});
```

Funktionen, die anonym als Parameter übergeben werden, um unter bestimmten Bedingungen ausgeführt zu werden, werden als „Callback-Funktionen“ bezeichnet. Die Funktionen werden wie Variablen behandelt.

# 12.Stylesheet-Einstellungen ändern

Jedes Element hat eine „style“-Eigenschaft.

Wichtig ist hierbei, dass die Stylesheet-Notation in abgewandelter Form verwendet wird. Was in CSS mit einem „-“ geschrieben wird, erhält statt dessen einen Großbuchstaben.

| | |
|---|---|
| width | → width |
| text-decoration | → textDecoration |
| background-color | → backgroundColor |
| border-bottom-color | → BorderBottomColor |

```
var element = document.getElementById('die-kennung');
element.style.backgroundColor = '#FFAABB';
```

Den Klassennamen eines Elementes ändern

```
element.className = 'produkt-link';
```

Attribute eines Elements ändern

```
var logo = document.getElementById('logo');
logo.src = 'images/bild.jpg';
logo.alt = 'Ein Bild';
```

Finden aller Links mit der Klasse „produkt-link“

```
var links = document.getElementsByTagName('a');
var result = [];
for (var i=0; i < links.length; i++) {
   if(links[i].className === 'produkt-link') {
     result.push( links[i] );
   }
}
```

HTML Inhalt eines selektierten Elements auslesen / setzen

```
var element = document.getElementById('die-kennung');
var htmlInhalt = element.innerHTML;
// ersetzen
element.innerHTML = "<strong>Ersetzt</strong>";
// anhängen
element.innerHTML += "<strong>Angehängt</strong>";
```

# 13.Eine CSS-Klasse hinzufügen

```
var element = document.getElementById('die-kennung');
var classes = [];
if(element.className) {
  // es könnten mehrere Klassen angegeben sein
  classes = element.className.split(" ");
}
for (i in classes) {
      var className = classes[i];
      if (className === "die-kennung") {
            delete classes[i]; // entfernen falls vorhanden
      }
}
classes.push("die-kennung");
element.className = classes.join(" ");
```