



Einführung in die Programmierung

- Google Apps Script & JavaScript -

Claudia Scheffler

13. September 2013



1 Array

2 Collections

3 Funktion



- 1 Array
- 2 Collections
- 3 Funktion



- 1 Array
- 2 Collections
- 3 Funktion



Fragen ?



Array

Was sind Arrays?



Array

Ein Array ist eine zusammengesetzte Datenstruktur, die mehrere Werte in getrennten und direkt adressierbaren Elementen zusammenfasst, um so thematisch zusammengehörende Variablen in einer Sammlung zu verbinden.



Array

Was sind Arrays?



- mit Arrays ist es möglich, mehrere Werte zu einer Liste zusammenzufassen.
- die einzelnen Werte, **Array-Elemente**, werden durchnummeriert oder mit Namen versehen und sind dadurch erreichbar
- **Vorteile:**
 - mit Algorithmen ist es möglich auf alle Array-Elemente zuzugreifen
 - können jederzeit um neue Elemente erweitert werden



Array

Was sind Arrays?



0	1	2	3	Index
Klaus	Mona	Hans	Lena	Daten

■ Speicherung folgender Datentypen:

- Strings
- Zahlen
- booleche Werte
- Objekte
- Arrays



Array

Elemente von Arrays



- die Deklaration einer Variable hat ausgereicht, um einen Wert in ihr zu speichern
- bei einem Array, muss nach der Deklaration zusätzlich noch der Konstruktor aufgerufen werden

Syntax

- `nameDesArray = new Array();`
- `nameDesArray = new Array(Anzahl der Elemente);`
- `nameDesArray = new Array(Element1, Element2, ..., Element n);`



Array

Elemente von Arrays



Zugriff auf die einzelnen Elemente

- `nameDesArray[index] = wert;`
- `einfacheVariable = nameDesArray[index];`



Beispiel

```
var city;  
city = new Array("Leipzig", "Berlin", "Dresden", "Görlitz",  
"Zittau");  
var city0 = city[0];  
var city1 = city[1];  
var city2 = city[2];  
var city3 = city[3];  
var city4 = city[4];
```

Welche Städte sind in den einzelnen Variablen (city0 - city4) gespeichert?



Array

Definierte Anzahl von Array-Feldern



- das Beispiel eignet sich, wenn die Anzahl der in einem Array bereitzustellenden Elementen schon bekannt ist, nicht jedoch die zu erfassenden Werte

Beispiel

```
measurementSeries = new Array(4);  
measurementSeries[0] = 1.2122;  
measurementSeries[1] = 1.2589;  
measurementSeries[2] = 1.2733;  
measurementSeries[3] = 1.1968;
```



Array

Freie Anzahl von Array-Elementen



- bei der Erzeugung des Arrays sind weder die Zahl der Elemente noch im Array zu speichernden Werte bekannt

Beispiel

```
freeTime = new Array();  
freeTime[0] = "Lesen";  
freeTime[1] = "Reiten";  
freeTime[2] = "Schwimmen";
```



Array

Ausgabe von Array-Elementen



- die Nutzung der Werte eines Arrays erfolgt wie bei der Zuweisung
 - durch Angabe des Array-Namens
 - durch Angabe der Laufnummer des gewünschten Elements umschlossen von zwei eckigen Klammern

Tipp!

Das erste mögliche Element hat immer die Laufnummer 0, unabhängig davon, ob erst bei einer späterem Laufnummer mit dem Zuweisen von Werten begonnen wurde.

- den Wert des Elements kann direkt im Dokument ausgeben oder auch einer Variablen zugewiesen werden



Array

Ausgabe von Array-Elementen



Beispiel

- Direkte Ausgabe:
 - `document.write(nameDesArray[index]);`
- Zuweisung des Wertes an eine Variable;
 - `var wert = nameDesArray[index];`



Array

Arbeiten mit Arrays



nameDesArray.length	Anzahl der Elemente im Array
nameDesArray.sort()	Sortieren der Elemente im Array
nameDesArray.reverse()	Umkehren der Reihenfolge der Elemente im Array
nameDesArray.join()	Verknüpfung aller Elemente zu einer Zeichenkette
nameDesArray.slice()	Kopieren eines gewünschten Bereichs aus dem Array



Array

Assoziative Arrays



- eine weitere Variante der Konstruktion und Abfrage von Arrays

Assoziatives Array

Ein Assoziatives Array kennzeichnet sich dadurch aus, dass die Array-Elemente nicht durch Indizes, sondern durch selbst gewählte, sich am Inhalt orientierte Namen geben und darüber angesprochen werden.

- Ausgabe oder Veränderung der Werte der Array-Elemente erfolgt ebenfalls über die Angabe des Namens



Array

Assoziative Arrays



Beispiel

```
var marks = new Array();  
marks["Klaus"] = 1;  
marks["Mona"] = 5;  
marks["Hans"] = 2;  
marks["Lena"] = 4;
```

Tipp!

Bei der Wahl der Namen für die Elemente sollte immer auf eine gleich bleibende Schreibweise geachtet werden. Schon unterschiedliche Groß- und Kleinschreibung führt dazu, dass statt einem Array-Element zwei angelegt werden.



Array

Mehrdimensionale Arrays



- ineinander verschachtelte Arrays
- Verwendung immer dann, wenn mehrere Sätze zusammengehöriger Daten in einem Array gespeichert werden sollen

Beispiel

```
var person = new Array();  
person[0] = new Array();  
person[0]["Vorname"] = "Fritz";  
person[0]["Nachname"] = "Walter";  
person[0]["E-Mail"] = "frwa@example.com";  
person[1] = new Array();  
person[1]["Vorname"] = "Sepp";  
person[1]["Nachname"] = "Herberger";  
person[1]["E-Mail"] = "sehe@example.com";
```



Array

Mehrdimensionale Arrays



- Zugriff auf die Elemente erfolgt
 - durch Angabe des Array-Namens
 - durch Angabe der Laufnummern oder assoziativen Namen des gewünschten Elements der Dimension umschlossen von jeweils zwei eckigen Klammern

Beispiel

```
document.write(person[0]["Vorname"] + " und " +  
person[1]["Vorname"] + " sind Kollegen.<br>");
```



Fragen ?



Collections

Unter einer Collection versteht man ein Array, das vom Browser selbst erzeugt wird, um die verschiedenen Elemente eines HTML-Dokuments darzustellen.

- Beispiel:
 - Arrays von Formularen
 - Arrays von Bildern
 - Arrays von HTML-Frames
 - Arrays von Browser-Plug-ins



Warum werden Collections in Array-Struktur verwaltet?

- in einem HTML-Dokument können jeweils mehrere solcher Objekte auftreten
 - Formulare
 - Bilder
 - Frames
- Collections haben eins gemeinsam:
 - die Laufnummer der Elemente stehen in direktem Bezug zu ihrem Auftreten im HTML-Dokument



Collections

Zugriff auf die Collection mit Nummern



- erste Möglichkeit, um auf die einzelnen Elemente einer Collection zuzugreifen, bieten die Indizes

Beispiel

```
document.images[index]
```

- es sind nicht einzelne Werte gespeichert, sondern ganze **Objekte**

Beispiel

```
document.images[index].src
```



Collections

Zugriff auf die Collection mit Namen



- zweite Möglichkeit, um auf die einzelnen Elemente einer Collection zuzugreifen, bieten die Namen
- für jedes Element kann mit dem Attribute **name** einen Namen vergeben werden und Zugriffen werden

Beispiel

```
<html>
<head> <title>Beispiel Bild </title> </head>
<body>

<script type="text/javascript">
var breite = document.images["Beispielbild"].width;
var hoehe = document.images["Beispielbild"].height;
document.write("<br>Das Bild hat eine Größe von "
+ breite + " mal " + hoehe + " Pixel.");
</script>
</body>
</html>
```



Tipp!

Empfehlenswert ist es, den Zugriff über die mit Attributen festgelegten Namen von HTML-Elementen zu wählen. Ist die Veränderung der Position von Bildern oder anderen Seitenelementen notwendig, so wird eine Überarbeitung des gesamten Skriptes erspart.

Tipp!

Die vergebenen Namen sollten eindeutig und nicht mehrfach vorkommen.



Fragen ?



- bei der Programmierung von Anwendungen kommen immer wieder gleiche oder ähnliche Anweisungsabfolgen vor
- zur Vereinfachung und besseren Handhabung fasst man solche immer wieder benötigten Anweisungsblöcke zusammen



Funktionen



Funktion

Unter einer Funktion versteht man einen Programmcode, der nicht sofort ausgeführt wird, aber explizit auf- bzw. abgerufen werden kann. Diese führt entweder ein paar Befehle aus (Prozedur), oder es wird ein Wert zurückgegeben.

■ Aufgabe:

- Programmskripte zu "rationalisieren"



Syntax

```
function Funktionsname(Parameter1, Parameter2...) {  
  Programmblock;  
}
```

- beginnt mit dem Schlüsselwort **function** gefolgt vom **Namen** der Funktion
- innerhalb der runden Klammern befindet sich die **beliebige Anzahl von Parametern** der Funktion → existieren keine Parameter so bleibt die Klammer leer
- der Programmblock der ausgeführt werden soll, befindet sich in einem Paar geschweifter Klammern



Funktion

Was ist eine Funktion?



- Anweisungen in Funktionen werden nicht sofort ausgeführt, wenn die Funktionsdefinition vom Interpreter gelesen wird
- Anweisungsblock wird nur dann ausgeführt, wenn die Funktion direkt aufgerufen wird
- **Beispiel:**
 - Anklicken eines Buttons durch den Nutzer
- es existieren vordefinierte Funktionen und es können auch individuelle Funktionen programmiert werden



- Funktionen benötigen wie Variablen einen Namen
- damit sie aufgerufen werden können und um dem Programmieraufwand zu minimieren
- **Erlaubt ist:**
 - Buchstaben
 - Ziffern
 - Unterstrich (_)
- **Nicht erlaubt ist:**
 - Leer- und Sonderzeichen
 - das erste Zeichen ist eine Ziffer
 - JavaScript Schlüsselwörter und -begriffe



- eine Funktion wird aufgerufen, wenn im Quelltext der Name der Funktion mit dem Klammerpaar und evtl. vorhandenen Parameter dahinter geschrieben wird

Beispiel

- `alert();`
- `add(2,3);`
- `var summe = add(3,5);`

- die verwendeten Klammern kennzeichnen den Funktionsaufruf für den Interpreter



Funktion

Funktionen mit einem Rückgabewert



- in einigen Fällen ist es wichtig, dass die Funktion einen Wert zurück gibt → **Rückgabewert**
- **Beispiel:**
 - Ergebnis einer Berechnung
 - Booleschen Wert (true oder false)
 - usw.

Syntax

```
function myFunction() {  
  var x=5;  
  return x;  
}
```



Beispiel

```
function main(){  
  var x = 5;  
  var y = 4;  
  var sum = add(x, y);  
}  
function add(x, y){  
  var sum = x+y;  
  return sum;  
}
```

- der Interpreter ersetzt bei der Ausführung des Programms den Aufruf der Funktion durch den Rückgabewert



Funktion

Funktionen mit einem Rückgabewert



- **return** dient also dazu, den Wert innerhalb einer Funktion als Rückgabewert zu definieren
- die Ausführung der Funktion wird sofort beendet

Achtung!

Folgen nach dem Schlüsselwort weitere Anweisungen, werden diese einfach ignoriert!



Parameter

Unter einem Parameter versteht man einen Wert/e , die als Eingangsgröße beim Aufruf einer Funktion von dem aufrufenden Programm übergeben oder als Ergebnis von der Funktion an das aufrufende Programm zurückgegeben werden.

- können innerhalb der Funktion wie normale Variablen verwendet werden
- es muss sich dabei aber nicht mehr um die Deklaration und Initialisierung gekümmert werden



Beispiel

```
<html>
<head>
<title>Beispiel einer Funktion mit Werteübergabe</title>
<script type="text/javascript">
function ausgabe(ihr_name) {
alert("Ihr Name lautet: " + ihr_name);
}
</script>
</head>
<body>
<p>normales HTML</p>
<script type="text/javascript"> ausgabe("Hans Müller");
ausgabe("Gerda Meier");
</script>
<p>normales HTML</p>
</body>
</html>
```



eval()	Interpretiert einen übergebenen Wert als JavaScript-Code und führt ihn aus
parseInt()	Wandelt eine Zeichenkette in eine Ganzzahl um
parseFloat()	Wandelt eine Zeichenkette in eine Gleitkommazahl um
isNaN()	Überprüft, ob es sich beim Argument um einen nicht-numerischen Wert handelt
isFinite()	Überprüft, ob eine Zahl endlich ist
escape()	Kodiert Sonderzeichen in einer Zeichenkette
unescape()	Dekodiert mit escape() kodierte Sonderzeichen in einer Zeichenkette



Globale Variable

Eine Variablen, die **nicht** innerhalb einer Funktionen oder ohne das Schlüsselwort **var** definiert ist, wird globale Variable genannt.

- dieser Art von Variablen können überall im Quelltext verwendet werden

Lokale Variable

Eine Variablen, die innerhalb einer Funktionen definiert ist, wird lokale Variable genannt.

- diese Art von Variablen können nur innerhalb der Funktion in der sie definiert wurden sind, genutzt werden



Funktion

Lokale und globale Variablen in Funktionen



Beispiel

```

+-----> var myGlobal = 42;      // erzeugt eine globale Variable
myGlobal
|
|      function func1(arg1, arg2)
|      {
+----->   var myNum;           // erzeugt eine lokale Variable myNum
|           myNum = myGlobal + 100;
+----->   ...
|       }
|
|      function func2(arg1, arg2)
|      {
+----->   var myNum    = 200;  // erzeugt eine lokale Variable
myNum
|           myVar       = 50;   // ohne Schlüsselwort var:
|                               // erzeugt eine globale Variable
myVar
|           var myGlobal = 100; // überschreibt die globale
|                               // Variable myGlobal, aber nur
|                               // in diesem Block
+----->   ...
|       }
|
|      function func3(arg1, arg2)
|      {
+----->   ...
|       }

```



Fragen ?

Claudia Scheffler
siclsche@stud.hszg.de