



Einführung in die Programmierung

- Google App Script & JavaScript -

Claudia Scheffler

11. September 2013



1 Organisatorisches

2 Google-App-Script

3 JavaScript

4 Grundlagen

- Einbindung von JavaScript in HTML
- JavaScript in separaten Dateien
- Kommentare
- Variablen
- Operatoren
- Kontrollstrukturen



1 Organisatorisches

2 Google-App-Script

3 JavaScript

4 Grundlagen

- Einbindung von JavaScript in HTML
- JavaScript in separaten Dateien
- Kommentare
- Variablen
- Operatoren
- Kontrollstrukturen



1 Organisatorisches

2 Google-App-Script

3 JavaScript

4 Grundlagen

- Einbindung von JavaScript in HTML
- JavaScript in separaten Dateien
- Kommentare
- Variablen
- Operatoren
- Kontrollstrukturen



- 1 Organisatorisches
- 2 Google-App-Script
- 3 JavaScript
- 4 Grundlagen
 - Einbindung von JavaScript in HTML
 - JavaScript in separaten Dateien
 - Kommentare
 - Variablen
 - Operatoren
 - Kontrollstrukturen



Sommerkurs

Kompletter Sommerkurs



- **Zeitraum:** 11.09.2019 - 24.09.2013
- **Beginn:** 09:00 Uhr
- **Ende:** 15:00 Uhr



- 90 Minuten Vorlesung
- 30 Minuten Pause
- 90 Minuten Übung → Gern auch länger!



Meine Person

Claudia Scheffler



- studiere seit 2009 an der Hochschule Zittau/Görlitz Informatik
- 2009-2012 Bachelor-Studium
- seit 2012 Master-Studium
- **Kontakt:** siclsche@stud.hszg.de



Fragen ?



Google Apps Script

Was ist Google Apps Script ?



- Cloud-basierte Skriptsprache zur Entwicklung von leichtgewichtigen Anwendungen in der Google Apps-Plattform
- basiert auf JavaScript
- serverseitig in der Google-Cloud ausgeführt
- Erweiterung von Google-Apps:
 - Spreadsheets (Tabellen)
 - Dokumente
 - Google-Sites
- Automatisierung und zeitgesteuerte Ausführung von Aufgaben



Fragen ?



JavaScript

Was ist JavaScript?



- Programmiersprache des Internets & wurde für Webseiten entwickelt
- Interaktionen in statische HTML-Seiten und ist die Grundlage für komplexe Anwendungen
- reduziert den aufwändigen Transfer zwischen Benutzer, Browser und Server
- ereignisgesteuerte Sprache und erkennt Ereignisse
- nutzt den Browser als Benutzeroberfläche
- überlässt dem Browser die komplexen Aufgaben der Programmierung



■ **Anfangs:**

- Effekte
- Validierung von Formularen

■ **Heute:**

- der Kanal für die direkte Kommunikation mit dem Benutzer
- erlaubt dem Benutzer, mit der Webseite in Interaktion zu treten
- d.h den Inhalt der Seite zu ändern, nach dem die Seite im Browser geladen wurde
- heute ist die Interaktion zwischen dem Benutzer und dem Inhalt die Grundlage des Internets



Scriptsprachen

Scriptsprachen sind Programmiersprachen, deren Ziel es ist, nur kleinere Anwendungen oder Anweisungsfolgen zu realisieren. Dabei weisen die Scripte bestimmte spezifische Merkmale auf. Eines dieser Merkmale ist es, das Scripte vielfach nicht von einem Compiler in maschinenlesbaren Code übersetzt werden, sondern zur Laufzeit von einem Interpreter ausgeführt werden. Scriptsprachen sind oft auch spezifischen Aufgaben zugeordnet, so dass deren Syntax leicht zu überschauen ist.



■ Mit einfachen Worten:

- die Auswertung findet Zeile für Zeile statt
 - Anweisungen müssen nicht in eine Maschinensprache übersetzt werden
 - **Nachteil:** kleinere Fehler können erst dann entdeckt werden, wenn das Script ausgeführt wird
- bei Programmiersprachen wie Java oder C
- muss zuerst in eine maschinennahe Sprache kompiliert werden
 - kann anschließend ausgeführt werden



JavaScript

JavaScript, Java und JScript



- ursprünglich als "LiveScript" von Netscape entwickelt und später in JavaScript umbenannt
- lehnt sich an die Idee der Programmiersprache C und Java an
- nach dem Erfolg von JavaScript in Netscape entwickelte Microsoft JScript
- Antwort darauf war die Standardisierung von JavaScript als ECMAScript
- durch die Entwicklung von JavaScript und JScript sind beide Scriptsprachen den neueren Versionen von ECMAScript immer näher gekommen → konform zum Standard ECMAScript 262 v3



JavaScript

JavaScript & ECMAScript



- ECMA legt den Kern der JavaScript-Spezifikation fest
- bei webbasierten Anwendungen ist das Konsortium W3C verantwortlich
- das DOM legt Empfehlungen für Zugriff auf Dokumente, Stylesheets, das Fenster des Browsers und das Event Handling vor
- da sich alle modernen Browser dem DOM des W3C verpflichtet haben, ist die plattformübergreifende Programmierung mit Javascript einfacher geworden



Fragen ?



- JavaScript-Code wird direkt im HTML-Quelltext eingebettet und vom Browser interpretiert
- der Code wird wie HTML im Klartext übertragen und ist somit für den Menschen lesbar

Syntax

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



Einbindung von JavaScript in HTML



Beispiel

Beispiel

```
<html>
<head>
<title>Mein erstes JavaScript-Programm</title>
</head>
<body>
<p>normales HTML</p>
<script type="text/javascript">
document.write("Hallo JavaScript");
</script>
<p>normales HTML</p>
</body>
</html>
```



Beispiel

```
<script type="text/javascript">  
document.write("Ich lerne gerne JavaScript");  
</script>
```

- im Attribut type des script-Elements wird der MIME-Type (Multipurpose Internet Mail Extensions) des im Element enthaltenen Scriptes angegeben
- Kennzeichnung, durch die der Computer feststellen kann, um welche Art von Dokumentenabschnitt es sich handelt



Fragen ?



JavaScript in separaten Dateien



- Anweisungen oder Funktionen die von mehreren HTML-Dokumenten genutzt werden
- einzubinden über die **src**- und **type**-Attribute des **script**-Tags

Einbindung der separaten Datei

```
<script src="test.js" type="text/javascript">
```



JavaScript in separaten Dateien

Beispiel



Beispiel

```
<html>
<head>
<title>Testdatei</title>
</head>
<body>
<p>normales HTML</p>
<script src="test.js" type="text/javascript">
</script>
<p>normales HTML</p>
</body>
</html>
```



Fragen ?



- ein Script kann mit kommentierten Text versehen werden, der vom Interpreter ignoriert wird
- **Vorteil:**
 - Programmcode wird nachvollziehbar und strukturiert
 - bestimmte Abläufe auch im Nachhinein noch verständlich auszudrücken
 - bei der Zusammenarbeit von mehreren Personen

Syntax für einzeiligen Kommentar

```
// Kommentartext
```

Syntax für mehrzeiligen Kommentar

```
/*  
* Kommentartext  
*/
```



Fragen ?



Variablen

Was ist eine Variable?



Variable

Eine Variable ist ein "Behälter" zur Speicherung von Information.



- Variablennamen bestehen aus einer Folge von:
 - Buchstaben
 - Ziffern
 - Unterstrichen (_)
- Variablennamen beginnen entweder mit einem Buchstaben, Dollarzeichen (\$) oder Unterstrich (_)

Achtung!

Das erste Zeichen darf jedoch keine Ziffer sein!



- Variablennamen sind **Case Sensitiv**

Case Sensitiv

- Unter Case Sensitiv versteht man die Beachtung der Groß- und Kleinschreibung der Daten.
- **Beispiel:** *googleScript*, *GoogleScript*, *googlescript*
- Schlüsselwörter und -begriffe dürfen nicht als Variablennamen verwendet werden



Variablendeklaration

Die Deklaration bezeichnet die Vereinbarung eines Namens für einen Speicherplatz des Rechners. Der Programmierer teilt also dem Rechner mit, dass er in seinem Programm eine Variable benötigt.

- Deklaration einer Variable beginnt mit dem Schlüsselwort **var**

Beispiel

```
var city;  
var x;
```



Variablen

Variablendeklaration



- Wertzuweisung einer Variable erfolgt über das Gleichheitszeichen =

Beispiel

```
city = "Görlitz";  
var y = 5;
```



Variablen

Variableninitialisierung



Variableninitialisierung

Der Begriff Initialisierung bezeichnet die erste Zuweisung eines Wertes an eine Variable.

- ein häufiger Fehler in der Programmierung, ist das nicht initialisieren einer Variable



Variablen

Datentypen



- existiert nur ein kleiner Satz von primitiven Datentypen
 - Boolean
 - Number
 - String
- spezielle Werte sind
 - null
 - undefined
- alles andere sind Arten vom Typ Object



- mit dem Schlüsselwort **cons** werden Konstanten angelegt
- können gelesen, aber nicht verändert werden
- dienen dazu, einen gleichbleibenden Wert nutzen zu können

Beispiel

- `const mwst = 16;`
- `const a = 'b';`
- `const pi = 3.14159,v=4,c=88.8888;`



Variablen

Literale



- stellen feste Werte dar, die dem Script übergeben werden
 - Array-Literal `var farben = ["rot", "grün", "blau"];`
 - Boolean-Literal `var isln = true` und `var isln = false`
 - Floating-Point Literal `var pi = 3.1415`
 - Integer-Literal `var antwort = 42`
 - Objekt-Literal `var adress = { name: "Hansen", ort: "Bonn", plz: 4711 }`
 - String-Literal `var lamm = "bähbäh";`



Fragen ?



Operatoren

Arten von Operatoren



Operatoren

- Durch Operatoren werden Variablen miteinander kombiniert.
- Arten:
 - Arithmetische Operatoren : $+$, $-$, $*$, $/$, $\%$, $\neg$
 - Boolesche Operatoren : $\&\&$, $||$, $!$
 - Vergleichsoperatoren : $==$, $!=$, $<$, $>$, $<=$, $>=$
 - String-Operatoren



Fragen ?



Kontrollstrukturen

Eine Kontrollstruktur ist ein wesentliches Element zur Strukturierung von Computerprogrammen und regelt, in welcher Abfolge die im Quelltext aufgeführten Befehle abgearbeitet werden.

- Kontrollstrukturen in JavaScript:
 - For-Schleife
 - While-Schleife / Do-while-Schleife
 - If-Anweisung
 - Switch-Case-Anweisung



- diese Art von Schleife führt eine Gruppe von Anweisung (Block) mit einer bestimmten Anzahl von Wiederholungen aus

Syntax

```
for (Initialisierung; Bedingung; Befehlsfolge)
{
  Anweisung;
}
```



■ Initialisierung:

- Anweisung in dem der Schleifenzähler definiert und initialisiert wird
- **Mit einfachen Worten:** Zählvariable, die die Anzahl der Wiederholungen zählt

■ Bedingung:

- Die Schleife wird solange ausgeführt, bis diese Bedingung nicht mehr erfüllt ist.

■ Befehlsfolge:

- Nach jedem Durchlauf der Anweisung wird diese Befehlsfolge ausgeführt



Beispiel

```
for (var i=0; i<5; i++)  
{  
  x=x + "Die Nummer ist " + i + "<br>";  
}
```

Ergebnis:

Die Nummer ist 0.
Die Nummer ist 1.
Die Nummer ist 2.
Die Nummer ist 3.
Die Nummer ist 4.



Fragen ?



- die While-Schleife dient dazu, eine Abfolge von Anweisungen mehrfach auszuführen, solange eine Bedingung erfüllt ist
- die Überprüfung der Bedingung erfolgt vor der Abarbeitung des Anweisungsblockes

Syntax

```
while (Bedingung)
{
  Anweisungsblock;
}
```



Beispiel

```
var x="";  
var i=0;  
while (i< 5)  
{  
x=x + "Die Nummer ist " + i + "<br>";  
i++;  
}
```

Ergebnis:

Die Nummer ist 0.
Die Nummer ist 1.
Die Nummer ist 2.
Die Nummer ist 3.
Die Nummer ist 4.



- die Do-While-Schleife ist eine Variante der While-Schleife
- der Unterschied zur While-Schleife:
 - Anweisungsblock wird einmal ausgeführt und anschließend wird überprüft, ob die Bedingung erfüllt ist
 - danach wird die Schleife solange wiederholt, bis die Bedingung nicht mehr korrekt ist

Syntax

```
do  
{  
  Anweisungsblock;  
} while (Bedingung);
```



Beispiel

```
var x="";  
var i=0;  
do  
{  
x=x + "Die Nummer ist " + i + "<br>";  
i++;  
}  
while (i<3);
```

Ergebnis:

Die Nummer ist 0.

Die Nummer ist 1.

Die Nummer ist 2.



Fragen ?



Kontrollstrukturen: Fallunterscheidung

If-Anweisung

- eine If-Anweisung wird unter bestimmten Anweisungen ausgeführt
- die Verzweigung legt fest, welcher von zwei oder mehreren Anweisungsblöcken, abhängig von einer oder mehreren Bedingungen ausgeführt wird

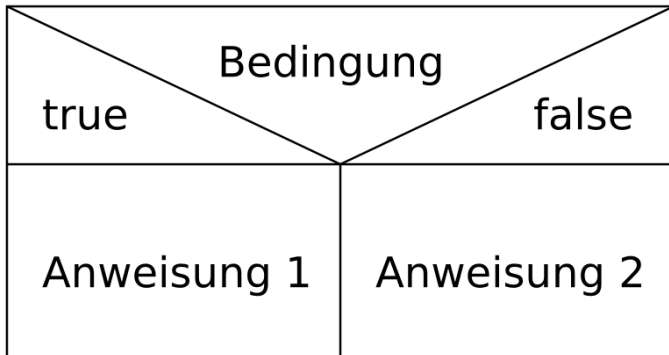
Syntax

```
if (Bedingung)
{
Anweisungsblock;
} else {
Anweisungsblock;
}
```



Kontrollstrukturen: Fallunterscheidung

If-Anweisung





Beispiel

```
var x="";  
var time=new Date().getHours();  
if (time < 20)  
{  
  x="Guten Tag";  
} else {  
  x="Guten Abend";  
}
```

Ergebnis:

Zeit vor 20 Uhr	Zeit nach 20 Uhr
Guten Tag	Guten Abend



Kontrollstrukturen: Fallunterscheidung

Switch-Case-Anweisung

- Mithilfe dieser Anweisung ist es möglich mehr als nur zwei Unterscheidungen, wie bei der If-Anweisung, zu differenzieren

Syntax

```
switch (Ausdruck){  
  case Wert1:  
    Anweisungsblock;  
    break;  
  case Wert2:  
    Anweisungsblock;  
    break;  
  default:  
    Anweisungsblock;  
}
```



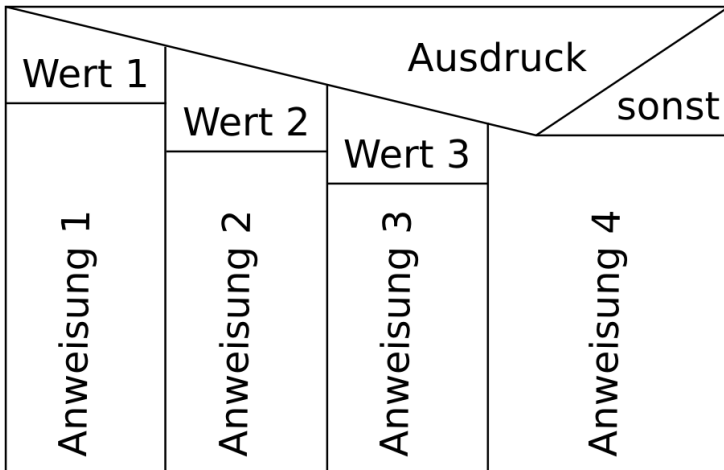
Achtung!

Das Schlüsselwort **break** darf am Ende der Switch-Case-Anweisung nicht fehlen, sonst wird die Überprüfung des Ausdrucks im passenden Fall nicht abgebrochen.



Kontrollstrukturen: Fallunterscheidung

Switch-Case-Anweisung





Kontrollstrukturen: Fallunterscheidung

Switch-Case-Anweisung

Beispiel

```
var x;  
var d=new Date().getDay();  
switch (d)  
{  
  case "Sonntag":  
    x="Heute ist Sonntag";  
    break;  
  case "Montag":  
    x="Heute ist Montag";  
    break;  
  case "Dienstag": x="Heute ist Dienstag";  
    break;  
  case "Mittwoch":  
    x="Heute ist Mittwoch";  
    break;  
  case "Donnerstag":  
    x="Heute ist Donnerstag";  
    break;  
  case "Freitag":  
    x="Heute ist Freitag";  
    break;  
  case "Samstag":  
    x="Heute ist Samstag";  
    break;  
}
```



Fragen ?

Claudia Scheffler
siclsche@stud.hszg.de