JavaScript

Einheit 01

Einführung

Was ist JavaScript?

- Programmiersprache
- Läuft im Browser oder eingenstandgig als nodeJS)Unterstützung für Windows, macOS und Linux)
- Interpreter-Sprache (Skriptsprache)

Exkurs Programmiersprachen

Exkurs: Programmiersprachen

> Eine Programmiersprache ist eine formale Sprache zur Formulierung von Datenstrukturen und Algorithmen, d. h. von Rechenvorschriften, die von einem Computer ausgeführt werden können. Sie setzen sich aus Anweisungen nach einem vorgegebenen Muster zusammen, der sogenannten Syntax.

Wikipedia, 16.10.18

Exkurs: Programmiersprachen (1/2)

Unterscheidung von Code in

- Maschinencode
- Assembly
- Programmcode

Exkurs: Programmiersprachen (2/2)

Unterscheidung nach Art der Ausführung

- Compiler
- Interpreter

Unterscheidung nach Ort der Ausführung

- Nativ
- Laufzeitumgebung

Beispiele für Programmiersprachen

- C, C++, C#
- Java
- .NET
- Python
- Perl, PHP, Ruby
- Javascript
- R
- COBOL, FORTRAN

https://www.tiobe.com/tiobe-index/

Die Programmiersprache JavaScript

JavaScript im Detail

- Web-Skriptsprache
- Läuft direkt im Browser
- interpretiert, Objekt-basiert
- dynamische Typisierung
- Keine Beziehung zu Java (!!!)
- Verwendung mit node.js auch am Server möglich

Geschichte

- Entwickelt von Netscape 1995 (Browserverion 2) Unterstützung in Internet Explorer ab Version 3
- Standardisiert als ECMAScript durch die ECMA (European Computer Manufacturers Association)
- Moderne Browser weitgehend kompatibel zum aktuellen ECMA-Standard Aktuell 8th Edition ECMAScript 2017 - kurz ES.8

Anwendungsbereiche (1/2)

- Traditionelle Beispiele
  - Formulareingaben prüfen
  - Fehlermeldung anzeigen, Bestätigungen abfragen
- Moderne Beispiele
  - Universelle Softwarebasis für plattformübergreifende Anwendungen
  - Moderne Webanwendungen setzen verstärkt auf JavaScript und haben alle anderen Ansätze verdrängt (z.B. Adobe Flash)
  - Entwicklung von Mobile Apps

Anwendungsbereiche (2/2)

- Funktionsumfang
  - Vollwertige Programmiersprache
  - Verarbeitung von Maus- und Tastatureingaben
  - Dynamische Erzeugung von (HTML-)Ausgabe
  - Zugriff auf Dokument-Struktur über das Document Object Model (DOM)

Einbinden von JavaScript (1/2)

In HTML

```
<script>
        alert('Hello JavaScript!');
</script>
```

Direkt in HTML Element

```
<a href="javascript: alert('Hallo Welt');">Click!</a>
<a onclick="javascript: alert('Hallo Welt');">Click!</a> <!-- Event Handler-->
```

Einbinden von JavaScript (2/2)

Externe Datei

```
<script src="hello.js">!</script>
```

Beispiel zur Einbindung

- `11_einbindung_script_alert.html`
- `12_einbindung_script_document_write.html`
- `13_einbindung_onclick.html`
- `14_einbindung_external.html`

Variablen

In JavaScript kann man Zahlen und Werte - wie in jeder Programmiersprache - in Variablen speichern, die zur Laufzeit des Programms aufgerufen werden können.

```
let meine_zahl = 3;
let mein_sting = 'Hallo Welt!';
```

Mein erstes JavaScript

Einfügen der folgenden Passage in ein HTML Dokument.

```
<script>
  alert("Hallo Welt")
</script>
```

Datei: `01_hallo_welt.html`

Erste Funktionen

```
alert("Hallo")
document.write("Welt")
console.log("!")
```

Kommentare

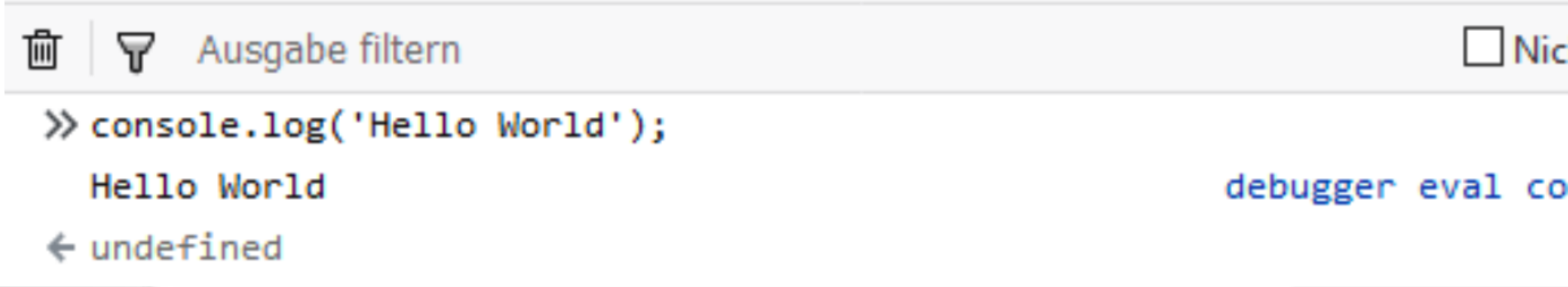
Kommentare sind Teile des Programms die nicht ausgeführt werden.

```
// Einzeiliges Kommentar

/* Kommentar über
mehrere Zeilen */
```

Browser Console

- Entwicklerkonsole des Browsers zur Laufzeitumgebung
- Direkte Eingabe in die Konsole möglich
  - zB `console.log('Hello World');`



Einfacher Zugriff auf HTML Elemente

```
<script>
  // Allgemein
  document.getElementById()
  document.getElementById('id').value

  // Formulare (einfach für den Beginn)
  document.forms.Testformular.elements.Eingabe.value
</script>
```

Dazu

- `21_zugriff_auf_HTML_elemente.html`

Exkurs Strings in Zahlen umwandlen

```
let zahl = document.getElementById('zahl').value

let echteGanzZahl = parseInt(zahl)
let echteKommaZahl = parseFloat(zahl)
```

Übung 01

Datei: `02_hallo_user.html`

```
<input type="text" id="name">
<button onclick="alert('Hallo ' + document.getElementById('name').value)">Click!</button>
```

Übung: Taschenrechner

Wir wollen einen kleinen Taschenrechner für Additionen erstellen.

Dazu erstellen wir das die Datei `03_additions_taschenrechner.html`. Hierbei sollen wir dem Nutzer zwei Formularfelder angezeigt werden und wir berechnen auf Knopfdruck das Ergebnis.

Grundlagen der Programmierung

In JavaScript

Variablen und Operatoren

Variablen dienen zur Speicherung von Werten

```
let a = 3
let b = 5
```

Operatoren dienen zur Ausführung von Berechnungen

```
c = a + b
```

Operatoren: `+` `-` `*` `/` `%` `=`

Einfache Datentypen

- Zahlen (int, float, ...)
  - Ganzzahlen: 0, 22, -1000
  - Fließkommazahlen (Punkt als Trennzeichen): 33.333, 123.0
- Zeichenketten (string)
  - Beliebig lange Folgen von Zeichen (können durch + aneinander gekettet werden)
- Wahrheitswerte (boolean)
  - true, false
- Sonderwerte
  - undefined: Nicht initialisierte Variable, Funktion ohne Rückgabewert, ...
  - null: Spezielles Objekt ohne Wert
  - NaN: “not a number” - Nicht-nummerischer Wert in nummerischem Kontext

Schwache Typisierung

Jede Variable und jeder Funktionsparameter kann uneingeschränkt Werte jedes JavaScript-Datentyps annehmen

- Zahl (Ganzzahl, Fließkomma), Zeichenkette, Wahrheitswert, Array, Objekt, Funktion

Exkurs: Zufallszahlen

```
  const max = 3 // Maximal Zahl
  const random = Math.floor(Math.random() * max);

  // oder

  const max = 3 // Maximal Zahl
  const random = Math.ceil(Math.random() * max);
}
```

Hierzu: `04_zufallszahl.html`

JavaScript

Einheit 02

Verzweigungen

Einfache Verzweigungen

Verzweigungen oder sog. IF-Statements werden genutzt zu die Programmablauf anhand von Konditionen zu bestimmen.

```
if (1 == 1) {
    mach_was()
} else {
  mach_etwas_anderes()
}
```

Beispiel: Einfaches Kopfrechenspiel

Wir wollen ein Spiel erstellen in dem der Benutzer eine Addition lösen muss. Am Ende des Programms wird die richtige Antwort und die gegebene Antwort angezeigt. Ebenso soll mitgeteilt werden, ob die Antwort richtig oder falsch ist.

Datei: `05_Kopfrechenspiel.html`

Übung 01: Kopfrechenspiel

Wir wollen ein Spiel erstellen in dem der Benutzer eine Addition lösen muss. Die beiden Werte für die Addition sollen dabei von einem Zufallsgenerator erzeugt werden. Am Ende des Programms wird die richtige Antwort und die gegebene Antwort angezeigt. Ebenso soll mitgeteilt werden, ob die Antwort richtig oder falsch ist.

Als Basis kann das Beispiel `05_Kopfrechenspiel.html` verwendet werden.

Mehrfache Verzweigungen

```
if (1 == 1) {
    mach_was()
} if else (1 == 1){
  mach_etwas_anderes()
}
else {
  mac_etwas_ganz_anderes()
}
```

Mit `if else` kann eine weitere Bedingung definiert werden, die nach dem if ausgeführt wird. Anweisungen nach dem else werden nur ausgeführt, falls keine vorhergehende Bedingung zutraf.

Übung 02: Gehaltsberechnung (vereinfacht)

Der Anwender soll dazu aufgefordert werden sein monatliches Bruttogehalt einzugeben. Vom Bruttogehalt werden in jedem Fall 27,68 % für die Sozialversicherung abgezogen. Des Weiteren wird die Einkommenssteuer abgezogen: Bis 915 € ist keine Steuer zu zahlen. Darüber werden 25 % Steuer abgezogen. Die Ausgabe soll nach folgendem Schema erfolgen:

```
Brutto        1500,00
SV             415,20
Steuer         271,20
Netto          813,60
```

Datei: `02_Gehaltsberechnung_einfach.html`

Vergleichsoperatoren

Operator	Bedeutung
==	ist gleich
!=	ist ungleich
<	ist kleiner
>	ist größer
<=	ist kleiner gleich
>=	ist größer gleich

Exkurs: Boolesche Aussagenlogik

In der Booleschen Aussagenlogik gibt es nur zwei Werte

- WAHR (bzw. true oder 1)
- FALSCH (bzw. false oder 0)

Bei einem If-Statement wird auf den Booleschen Wahrheitswert (WAHR oder FALSCH) überprüft.

Logische Operatoren

`&&` Alle mit and verknüpften Bedingungen müssen wahr sein

`||` Eine der mit or verknüpften Bedingungen müssen wahr sein

`!` Der Operator not kehrt den Wahrheitswert um

Übung 03: Gehaltsberechnung

Der Anwender soll dazu aufgefordert werden sein monatliches Bruttogehalt einzugeben. Vom Bruttogehalt werden in jedem Fall 27,68 % für die Sozialversicherung abgezogen. Des Weiteren wird die Einkommenssteuer abgezogen: Bis 915 € ist keine Steuer zu zahlen. Bei einem Betrag **über 915 € bis 1785 € werden 25 % Steuer** abgezogen. Darüber werden **50 %** abgezogen. Die Ausgabe soll nach folgendem Schema erfolgen:

```
Brutto          1500,00
SV               415,20
Steuer (25%)     271,20
Netto            813,60
```

Datei: `03_Gehaltsberechnung.html`

Reihenfolge von Operatoren

- Vorzeichen: `+` `-`
- Rechenoperatoren: `*` `/` `%` `//`
- Rechenoperatoren: `+` `-`
- Vergleichsoperatoren: `>` `>=` `<=` `<` `==` `!=`
- `!`
- `&&`
- `||`

Schleifen

Schleifen

Neben Verzweigungen sind Schleifen eine weiter grundlegende Struktur in Programmen. Es gibt zwei Typen von Schleifen for : wird verwendet um Programmschritte für eine gewisse Anzahl zu wiederholen (Zählschleife) while : ist an eine Bedingung geknüpft (bedingungsgesteuerte Schleife)

For Schleife

```
for (let i = 0; i < 5; i++) {
  document.write("Zahl: ", i, ", Quadrat: ", i*i)
}

// break beendet eine Schleife
for (let i = 0; i < 5; i++) {
  if (i*i > 20) {
    break;
  }
  document.write("Zahl: ", i, ", Quadrat: ", i*i)
}
document.write('Ende');
```

Hierzu: `02_For_Schleife.html` und `03_For_Schleife.html`

Übung 04: Kopfrechenspiel mit 4 Versuchen

Der Benutzer bekommt eine Kopfrechenaufgabe und hat 4 Versuche für die richtige Antwort. Sofern der Benutzer richtig liegt wird das Ergebnis angezeigt und das Programm beendet. Nach Versuchen wird ebenso das Ergebnis angezeigt und das Programm beendet.

Datei: `04_Kopfrechenspiel_4_Versuche.html`

While-Schleife

```
// Init
let summe = 0

// while
while (summe < 50) {
   zahl = Math.floor(Math.random()*10)
   summe = summe + zahl
   document.write("Zahl:", zahl, "Summe:", summe)
}
```

Hierzu: `05_While_Schleife.html`

Übung 05: Kopfrechenspiel bis zum richtigen Ergebnis

Das Kopfrechenspiel soll solange ausgeführt werden, bis der Anwender das richtige Ergebnis eintippt.

Datei: `05_Kopfrechenspiel_while.html`

Übung 06: Umrechnung inch in cm

Ein Programm soll erstellt werden, das den Anwender wiederholt dazu auffordert mittels Dialogbox einen Wert in inch einzugeben. Der eingegeben Wert wird daraufhin in Zentimeter umgerechnet und ausgegeben.

1 inch = 2,54 cm

JavaScript

Einheit 03

Funktionen

Funktionen

Funktionen dienen zur Modularisierung (also Zerlegung) eines Programmes.

- Programmteile müssen nur einmal definiert werden
- Programmteile können wiederverwendet werden
- Umfangreiche Programm können in übersichtliche Teile zerlegt werden
- Pflege und Wartung wird erleichtert

Funktionen definieren

Wir definieren eine einfache Funktion zur Trennung unserer Ausgabe.

```
// Definition der Funktion
function sagHallo()
{
    alert('Hallo!');
}

sagHallo(); // Aufrufen der Funktion
```

Parameter

Funktionen können auch Übergabeparameter haben.

```
// Definition der Funktion
function quadrat(x)
{
    let q = x * x;
    console.log("Zahl: " + x + " Quadrat: " + q;)
}
```