JavaScript

für Maschinenbauer

Geschichtliches

- **JavaScript** (kurz **JS**) ist eine Skriptsprache, die ursprünglich 1995 von Netscape für dynamisches HTML in Webbrowsern entwickelt wurde, um Benutzerinteraktionen auszuwerten, Inhalte zu verändern, nachzuladen oder zu generieren und so die Möglichkeiten von HTML und CSS zu erweitern.[3] Heute findet JavaScript auch außerhalb von Browsern Anwendung, so etwa auf Servern und in Microcontrollern.[4][5]
- Der heutige Name, der ursprünglich **LiveScript** genannten Sprache entstand 1996 aus einer Kooperation von Netscape mit Sun Microsystems. Deren Java-Applets, erstellt mit der gleichfalls 1995 veröffentlichten Programmiersprache Java, wurden mit LiveScript, den Netscape Navigator integriert. Um die Popularität von Java zu nutzen, wurde LiveScript in JavaScript umbenannt, obwohl sich die beiden Sprachen stark voneinander unterscheiden.
- Der als **ECMAScript** (ECMA 262) standardisierte Sprachkern von JavaScript beschreibt eine dynamisch typisierte, objektorientierte, aber klassenlose Skriptsprache. Sie wird allen objektorientierten Programmierparadigmen unter anderem auf der Basis von Prototypen gerecht, deren Deklaration ab ECMAScript 6 mit einer Syntax ermöglicht wird, wie sie ähnlich auch bei klassenbasierten Programmiersprachen üblich ist. In JavaScript lässt sich je nach Bedarf objektorientiert, prozedural oder funktional programmieren.[6]

Was ist JavaScript (nicht)

Variable

- Speicher für Zahlen, Texte, Listen, Widgets, Objekte, …
- Benannte “Box”, in der beliebige Daten gespeichert werden können.
- Neue Variable mit `let` oder `var` definieren (Unterschied für uns irrelevant):

```
let age = 30
var name = "Johannes"
let isRotating = true
```

- Variable lesen

```
alert(name)
```

- Variable verändern

```
age = age + 1
isRotating = false
```

Kommentar

- Wird vom Browser ignoriert
- Sollen verwendet werden, um Codeteile zu beschreiben (“Warum”)
- Werden aber häufig verwendet, um (noch) nicht funktionierenden Code “auszublenden”
- Beginnen mit // und enden mit dem Zeilenende

```
// Diese Zeile wird vom Browser ignoriert ...
let name = "Johannes" // ... so wie dieser Text
```

Objekt

- Gruppe von Werten (“Eigenschaften”/”Properties”)
- Jeder Wert mit eindeutigem Schlüssel angesprochen
- Werte können Zahlen, Texte, usw., Funktionen oder wiederum Objekte sein

```
// Neues Objekt erstellen und in Variable speichern
let model = {
    name: "Propeller",
    weightKg: 5776.47,
    position: { x: -100, y: 20, z: 0 }
}
// Objekteigenschaft ändern
model.position.x = model.position.x + 10 // v1
model["position"]["x"] = model["position"]["x"] + 10 // v2
```

- v2 ist flexibler, weil Schlüssel auch Sonderzeichen enthalten können (z.B. “model-1”) und auch eine Variable als Schlüssel verwendet werden kann.

Funktion

- Führt beliebig viele Aktionen/Befehle/Anweisungen aus
  - Anweisungen werden bei jedem Aufruf ausgeführt
- Nur bedingt mit mathematischen Funktionen (z.B. sin) vergleichbar
- Kann beliebig viele Parameter definieren, die die Funktionsanweisungen "feinjustieren"
  - z.B. wie lange soll etwas animiert werden, welche Farbe soll eingestellt werden, um wie viel Grad soll etwas gedreht werden
- Kann optional ein Ergebnis berechnen (vergleichbar mit Ergebnis von math. Funktionen)

Funktion

- In Variable speichern

```
let animateXPosition = (widget, from, to) => {
    // Hier stehen die Anweisungen
}
// Funktionsaufruf
animateXPosition("model-1", 10, 100)
```

- In Objekt speichern

```
$scope.animateXPosition = (widget, from, to) => {
    // Hier stehen die Anweisungen
}
// oder
$scope["animateXPosition"] = (widget, from, to) => {
    // Hier stehen die Anweisungen
}
// Funktionsaufruf
$scope.animateXPosition("model-1", 10, 100)
```