
JS

JavaScript Basics

Dokumentinformationen

Titel:	JavaScript Basics
Thema:	JS lernen
Dateiname:	javascript-basics.pdf
Autor:	Sina Blattmann
Version:	1.0.0

Inhaltsverzeichnis

Zusammenfassung

Dieses Skript soll dir dabei helfen die Basics von JavaScript zu lernen. Wenn du dieses Skript durchgemacht hast, solltest du das Grundprinzip verstehen und ausserdem werde ich dich auch auf weitere Seiten verweisen, auf welchen du das ganze noch genauer nachlesen kannst. Natürlich wirst du noch viel Übung brauchen um JS zu beherrschen, dafür ist ein Grundgerüst nötig, welches ich dir in Form dieses Tutorials zur Verfügung stelle.

Vorkenntnisse

Um dieses Skript zu verstehen solltest du HTML schon etwas kennen. Ausserdem wäre es von Vorteil, wenn du schon einmal mit einer anderen Programmiersprache gearbeitet hättest. Aber auch wenn nicht, solltest du es schaffen die Dinge in diesem Skript zu verstehen. Falls etwas nicht klar ist, werde ich hier nun noch eine Seite angeben, auf welcher Dinge gut erklärt werden. Diese Seite wird dir hoffentlich weiterhelfen können, falls einige Dinge zu kompliziert erklärt wurden.

https://www.w3schools.com/js/default.asp

Für dein Verständnis würde ich dir raten das Skript der Reihe nach durchzulesen, da es aufbauend ist. Wenn du einmal das ganze Skript durchgelesen hast, kannst du immer wieder zu vorherigen Stellen zurückgehen.

Was ist JavaScript?

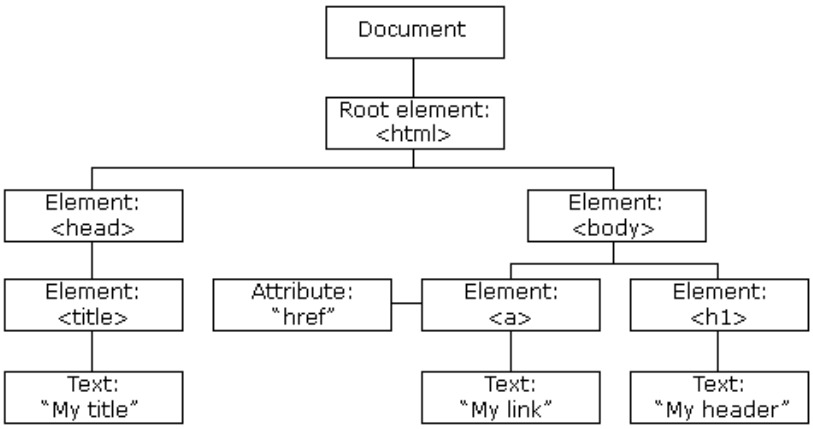
JavaScript (JS) ist eine Programmiersprache, welche, wenn man sie zu einem HTML Dokument hinzufügt, für Dynamik und Interaktion mit dem Nutzer sorgt. JS wurde von Brendan Eich erfunden.

JS wird wie bereits oben erwähnt zusammen mit HTML und CSS benutzt. JS kann HTML-Dokumente dynamisch verändern, wie es das genau macht, wird im nächsten Kapitel erklärt.

HTML-DOM

DOM steht für Document Object Model. Das HTML DOM ermöglicht es JS die verschiedenen HTML-Elemente einer Seite zu verändern. Beim Laden einer Webseite kreiert der Browser ein Document Object Model der Seite. Das HTML DOM wird als ein Baum von Objekten konstruiert.

Das oberste Element dieses Baums ist das Document, welchem alle verschiedenen HTML-Tags untergeordnet werden



Quelle: https://www.w3schools.com/js/js_htmldom.asp

Mit Hilfe dieses Models kann JS nun dynamischen HTML-Inhalt kreieren. Es kann alle möglichen Elemente auf der Seite bearbeiten, neue Elemente hinzufügen oder bereits existierende Elemente löschen.

Mit welcher IDE kann man JS programmieren?

Javascript kann von vielen verschiedenen IDE's verarbeitet werden. Eine IDE ist eine integrierte Entwicklungsumgebung, sie stellt die wichtigsten Werkzeuge zur Softwareentwicklung zur Verfügung. So können Programme einfacher erstellt werden. In einer IDE beinhaltet meistens einen Editor, ein Compiler, ein Debugger(wird benutzt um Fehler in einem Programm zu identifizieren) und auch noch viele andere Tools, welche dem Enwickler beim Programmieren behilflich sein können.

Eine sehr bekannte IDE, um JS zu entwicklen ist Visual Studio Code, welche es als Gratisversion gibt. Es gibt ausserdem auch noch Webstorm, welche ich persönlich auch noch eine gute Variante finde. Wenn man nun aber keine Lust hat eine IDE herunterzuladen, gibt es auch online viele Möglichkeiten, um gewisse Dinge mit JS auszuprobieren. Auf den folgenden Seiten kann HTML, CSS, und JS-Code kompiliert werden, was manchmal auch noch ziemlich nützlich ist.

https://jsbin.com/

https://codepen.io/pen/

JS mit HTML

Wohin kommt der JS-Code

In HTML der JS Code in *<script>* und *</script>* Tags eingefügt. Diese Tags können entweder im *<head>* oder auch im *<body>* platziert werden.

```
<script>
console.log("Hi there");
</script>
```

JS Skripts können aber auch in einem eigenen File geschrieben werden, welche mit der Endung *.js* enden soll. Danach können sie als Referenz im HTML-Dokument angegeben werden. Dazu muss ein *src* -Attribut im *<script>* Tag hinzugefügt werden.

```
<script src="myScript.js"></script>
```

Danach kann der Code dieser externen Datei genau gleich genutzt werden, wie wenn der Code direkt im HTML-File wäre.

Das Skript in einer externen Datei zu platzieren hat einige Vorteile. Es trennt das HTML und den JS-Code. Ausserdem ist es verständlicher die beiden zu lesen, denn wenn es ein riesiges Skript in der HTML-Datei hat, kann dies verwirrend sein.

JavaScript Output

Daten können von JS verschieden angezeigt werden. Ich werde in diesem Kapitel einige Arten von Output erläutern.

- innerHTML

  um ein bestimmtes HTML-Element zu bekommen kann die Methode *document.getElementById(id)* benutzt werden. Die *id* ist die Id des gewünschten Elements. Um dann den Inhalt dieses Elements zu ändernkann die *innerHHTML* property benutzt werden, denn sie definiert den HTML Inhalt.

```
document.getElementById(example).innerHTML = 'Hallo';
```

  Mit diesem Code Snippet wird nun also der Inhalt des HTML-Elements mit der Id *example* zu «Hallo» geändert.

- window.alert()

  Um eine Warnung anzuzeigen können Daten in einer alert box angezeigt werden.

```
window.alert('Achtung');
```

- console.log()

  Eine sehr oft gebraucht Funktion ist *console.log()*. Um zu prüfen ob gewisse Werte in einer Applikation stimmen, kann der Entwickler diese Funktion gebrauchen. Es ist sehr wichtig für das Debugging[1] eines Programmes. Die hier mitgegebenen Werte werden jedoch nicht auf der Seite selbst, sondern in der Konsole des Browsers angezeigt.

```
console.log("Hello World");
```

Variablen

Variablen sind Container, in welchen Werte gespeichert werden können. Diese können entweder mit dem Schlüsselwort *var* oder mit *let* deklariert werden. Natürlich soll nach diesem Schlüsselwort auch noch ein Name folgen, mit welchem die Variable identifiziert werden kann. Die Variablennamen können fast beliebig benennt werden, jedoch gibt es einige Restriktionen. Die Namen dürfen zum Beispiel keine Sonderzeichen enthalten.

Etwas weiteres, das beachtet werden soll, ist, dass JS die Gross-/Kleinschrift beachtet, *variablenName* und *variablenname* ist also nicht das gleiche. Eine weitere Regel ist, dass die Variablennamen in Camel Case geschrieben werden, was bedeutet, dass der Anfangsbuchstabe klein ist und alle darauffolgenden Wörter grossgeschrieben werden.

In JS muss ausserdem jede Befehlszeile mit einem Semikolon abgeschlossen werden, um das Ende dieser Zeile zu markieren. Wenn man diese Semikolons nicht macht läuft der Code trotzdem, jedoch können unerwartete Ergebnisse auftreten.

```
var variablenName;
let variablenName;
```

Nach der Deklarierung einer Variablen, kann ihr ein Wert zugewiesen werden.

```
variablenName = 1;
```

Oder diese beiden Schritte (Deklarierung und Wert zuweisen) können zusammengefasst werden.

```
let variablenName = 1;
```

Dieser Wert kann danach auch geändert werden, indem man der Variable einfach einen neuen Wert zuweist.

Datentypen

Variablen können verschiedene Datentypen sein.

String

Text, wird in Anführungszeichen gesetzt.

```
let string = 'James';
```

Number

Eine Nummer, nicht in Anführungszeichen

```
let number = 3;
```

Boolean

Ein true/false Wert, da true/false Schlüsselwörter sind keine Anführungszeichen.

```
let boolean = true;
```

Object

Kann alles Mögliche beinhalten, alles in JS ist ein Objekt und kann in einer Variable gespeichert werden

```
let object = document.getElementById('id');
```

Arrays

Arrays sind kein eigener Datentyp, sie gehören zu den Objekten. Jedoch sind sie erwähnenswert, da sie sehr oft gebraucht werden. In einem Array können mehrere Werte gespeichert werden. Hier nun ein Beispiel, wie ein Array kreiert warden kann.

```
let cars = ["Saab", "Volvo", "BMW"];
```

Inhalte von Arrays werden mit eckigen Klammern ernstellt. Etwas was zu beachten ist, dass die Daten in Arrays alle den Gleichen Datentyp haben, jedoch kommt es nicht darauf welcher das ist, man kann also sowohl Strings, als auch Zahlen oder Objekte ind Arrays abspeichern.

Um nun auf die einzelnen Elemente eines Arrays zugreifen zu können, macht man Gebrauch vom Index. Jedes Element hat ein Index, mit welchem es identifiziert werden kann. Das erste Element in einem Array hat immer den Index 0.

Um nun also das erste Element in eine Variable zu speichern würden wir wie folgt vorgehen.

```
let car = cars[0];
```

In unserem Fall wäre der String “Saab” nun in der Variable car gespeichert.

Operatoren

Addition -> **+**

```
let addition = 1 + 5;
```

Subtraktion -> **-**

```
let subtraktion = 17 - 14;
```

Multiplikation -> ***

```
let multiplikation = 5 * 7;
```

Division -> /

```
let division = 35 / 7;
```

Zuweisungsoperator -> mit dem **=** Zeichen wird ein Wert einer Variablen zugewiesen

Gleichheitsoperator -> prüft ob zwei Werte gleich sind und gibt einen Boolean, also true oder false, zurück

Es wird zwischen zwei unterschieden:

Bei dieser Variante wird geprüft, ob die beiden Variablen den gleichen Inhalt haben.

```
variable == 1
```

Bei der zweiten Variante wird geprüft, ob die beiden den gleichen Inhalt haben und der gleiche Datentyp sind.

```
variable === 1
```

Ungleich -> gibt true oder false zurück, prüft ob zwei Werte ungleich einander sind

```
variable !== 1
```
oder
```
!(variable === 1)
```

Kommentare

In JS können Kommentare entweder single-line oder mutli-line sein. Diese Kommentare werden vom Compiler nicht gelesen.

Single-line:

```
//single-line Kommentar
```

Multi-Line:

```
/*
Dies ist ein multi-line
Kommentar
*/
```

If/Else

Die if-Anweisung führt Anweisungen aus, wenn eine bestimmte Bedingung zu *true* ausgewertet wird. Wird die Bedingung zu *false* ausgewertet, können andere Anweisungen ausgeführt werden. Wenn man es auf Deutsch übersetzt, ist es einfacher zum verstehen: WENN etwas *true* ist dann wird etwas ausgeführt, SONST (wenn es nicht *true* ist) wird etwas anderes gemacht.

Hier nun ein kleines Codebeispiel, um zu sehen, wie das ganze aufgebaut ist.

```
let x = 3;
if (x > 5) {
  console.log("x ist grösser als 5.");
} else {
  console.log("x ist kleiner oder gleich fünf.");
}
```

In diesem Fall wurde x der Wert 3 zugewiesen. Nun wird also zuerst gecheckt ob x grösser als 5 ist. Da dies nicht der Fall ist wird die Funktion im *else* ausgeführt. In diesem Fall würde also die Nachricht «*x ist kleiner oder gleich fünf*» in der Konsole erscheinen.

For-Loop

Wie der Name schon sagt ist der For-Loop etwas, dass sich mehrmals wiederholt. Am einfachsten ist es, diese Thematik an dem Code selbst zu erklären.

```
for (statement 1; statement 2; statement 3) {
    // code block, der ausgeführt werden soll
}
```

Das Statement 1 wird ausgeführt, bevor der code block aufgerufen wird.

Das Statement 2 definiert eine Kondition, also wie oft der Codeblock ausgeführt werden soll.

Das statement 3 wird immer nach dem Codeblock ausgeführt.

Nun ein simples Beispiel:

```
for (i = 0; i < 5; i++) {
    console.log(i);
}
```

i wird am Anfang auf 0 gesetzt. Danach wird der Codeblock das erste Mal ausgeführt und es wird 0 ausgegeben, da i = 0 ist. Nach dem Code block wird i um eins erhöht, i ist nun 1 und bei diesem Durchgang wird also 1 ausegegeben, und so geht es dann weiter bis i nicht mehr kleiner als 5 ist.

Funktionen

Eine JS Funktion ist ein Codeblock, welcher ein bestimmtes Task ausführen muss. Eine Funktion wird ausgeführt, wenn sie aufgerufen wird. Um auch zu verstehen, wie eine Funktion aufgebaut ist, hier ein Beispiel.

```
function getLength(word) {
  return word.length;
}
```

Diese Funktion gibt die Länge eines Strings zurück wenn sie aufgerufen wird. Funktionen werden mit dem Keyword function erstellt, darauf folgt der Name der Funktion und Klammern, in welcher ein oder mehrere Parameter mitgegeben werden können. Diese Funktion könnte wie folgt aufgerufen werden.

```
getLength("hallo");
```

Aufgaben

https://www.w3schools.com/Js/js_exercises.asp

https://github.com/sinablattmann/js_tutorial