

## B.IV.8

### Algorithmen – Objektorientierte Programmierung

# Einheit: JavaScript-Grundlagen in HTML – Teil 1

Mirko Schiller



© RAABE 2024

© iStock/Getty Images Plus/panida wijitpanya

Um Besucherinnen und Besuchern einer Webseite ein „Gefällt mir“ oder „Teilen“ zu bieten, müssen Webentwicklerinnen und Webentwickler JavaScript beherrschen. In diesem Lerninhalt werden die Schülerinnen und Schüler in die Grundlagen der weltweit führenden Web-Entwicklungssprache JavaScript eingeführt. Sie lernen, wie man JavaScript in HTML einbettet. In diesem ersten Teil werden neben der Einführung in Funktionen auch Events vermittelt.

---

#### KOMPETENZPROFIL

<b>Klassenstufe:</b>	9/10, Sek. II
<b>Dauer:</b>	10 Unterrichtsstunden
<b>Lernziele:</b>	Die Lernenden ... 1. entwickeln auf Grundlage von JavaScript ein Verständnis für dynamische Benutzerinteraktionen, 2. implementieren algorithmische Grundstrukturen für Webseiten, 3. übertragen deren Kenntnisse auf andere Entscheidungsprozesse.
<b>Thematische Bereiche:</b>	Internet, HTML, JavaScript, Webentwicklung, interaktives Verhalten von Webseiten, Algorithmen, Kontrollstrukturen
<b>Kompetenzbereiche:</b>	Implementieren, Darstellen und Interpretieren, Produzieren und Präsentieren, Analysieren und Reflektieren



netzwerk  
lernen

zur Vollversion

## Auf einen Blick

### Benötigte Materialien

- PC/Laptop/mobiles Endgerät
- Stifte/Notizzettel



---

### Lektion 1

Thema: W-Fragen zu JavaScript: Was? Wozu? Wie? (90 Min.)

M 1 Wozu braucht man JavaScript?

M 2 Einbettung von JavaScript in HTML

M 2a Mein erstes JavaScript erwacht

- Benötigt:
- M 2a - Mein Erstes Javascript Erwacht.mp4
  - M2a.2.mein-erstes-javascript-erwacht-vorlage.zip

---

### Lektion 2

Thema: Basics in JavaScript (90 Min.)

M 3 Werte und Operatoren

M 3a Vertiefung: Datentypen

M 4 Deklaration und Initialisierung

---

### Lektion 3

Thema: Funktionen (90 Min.)

M 5 Funktionen in JavaScript

M 5a Übungen zum Thema JS-Funktionen

- Benötigt:
- M5a.uebung3-datum-vorlage.zip
  - M5a.uebung4-gradkonverter-vorlage.zip

---

### Lektion 4

Thema: Events & DOM (90 Min.)

M 6 Events in JavaScript

M 7 HTML bewusst nutzen, manipulieren oder interagieren

## Lektion 5

**Thema:** Zwischenfazit: Eingeben – Verarbeiten – Ausgeben (90 Min.)

**M 7a** **Eingeben – Verarbeiten – Ausgeben**

**M 7b** **Infokarte: Schrittfolge beim JavaScript-Handling**

**Benötigt:**  *M7b.JavaScript-Zusammenfassung.pdf*








### Benötigte Dateien

- Link:** Sammlung aller Programmcode-Vorlagen und Lösungen – [https://editor.p5js.org/mirkoschiller/collections/h\\_oQOYbLN](https://editor.p5js.org/mirkoschiller/collections/h_oQOYbLN)
- Video:** M 2a – Mein Erstes JavaScript Erwacht.mp4
- Programmcode:** M2a.2.mein-erstes-javascript-erwacht-vorlage.zip
- Programmcode:** M5a.uebung3-datum-vorlage.zip
- Programmcode:** M5a.uebung4-gradkonverter-vorlage.zip
- Infokarte:** M7b.JavaScript-Zusammenfassung.pdf

### Ergänzendes Material

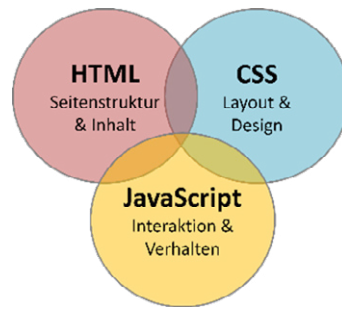
- Programmcode:** M5a.begrueessung-lsg.html
- Programmcode:** M5a.farbwechsel-lsg.html
- Programmcode:** M5a.uhrzeit-lsg.html
- Programmcode:** M5a.wochentag-lsg.html
- Programmcode:** M5a.zaehlen-lsg.html
- Programmcode:** M6a.1.span-lsg.html
- Programmcode:** M6a.2.alert-bei-mausschwebe-lsg.html
- Programmcode:** M6a.3.paragraphen-aendern-lsg.html
- Programmcode:** M6a.4.bild-vergroessern-lsg.html
- Programmcode:** M6a.5.nachricht-verlassen-lsg.html
- Programmcode:** M6a.5.nachricht-verlassen-lsg-alternative.html
- Programmcode:** M7.1-neuerDiv-lsg.html
- Programmcode:** M7.2-URL-lsg.html
- Programmcode:** M7a.1-taschenrechner-lsg.zip
- Programmcode:** M7a.2-rechtecksberechnung-lsg.zip
- Programmcode:** M7a.3-kreisumfang-lsg.zip

### Erklärung zu den Symbolen

	Dieses Symbol markiert differenziertes Material. Wenn nicht anders ausgewiesen, befinden sich die Materialien auf mittlerem Niveau.				
	einfaches Niveau		mittleres Niveau		schwieriges Niveau
	Zusatzaufgaben		Alternative		Selbsteinschätzung

## Wozu braucht man JavaScript?

Wenn wir eine Internetseite besuchen, besteht diese aus mehreren Elementen, die mit unterschiedlichen Programmiersprachen entwickelt wurden. In erster Linie besteht hierbei jede Webseite aus *HTML (HyperText Markup Language)*. Mit dieser Textauszeichnungssprache kann man auf einer Webseite Inhalte (Texte, Bilder, Videos usw.) geordnet strukturieren. Einfach gesprochen, ist HTML schwarze Schrift auf weißem Hintergrund in einer festgelegten Reihenfolge. Heutige Webseiten sind jedoch viel mehr. Um nun eine Webseite auch ästhetisch ansprechend zu gestalten oder gar bspw. mit ansprechenden Farben zu gestalten, wird *CSS (Cascading StyleSheet)* benötigt. Durch CSS kann man Webseiten also *formatieren*. Layout, Hintergründe und Farben von Elementen gestaltet man in einer CSS-Datei, die in der Regel für alle dazugehörigen HTML-Elemente Verwendung findet. Moderne Webseiten gehen jedoch noch einen Schritt weiter. Denn diese dienen auch dazu, Interaktionen der Besucherinnen und Besucher durchzuführen. Teilen, Senden oder Einloggen – es gibt viele Dinge, die man auf einer Webseite machen kann. Alle diese Tätigkeiten müssen von der Entwicklerin bzw. vom Entwickler vorher in der Webseite *implementiert* werden. Diese Programmierung geschieht mit *JavaScript* (kurz: *JS*). Diese Skriptsprache wurde in erster Linie für eben genau diese Webanwendungen entwickelt. Folglich kann JavaScript Benutzerinteraktionen auswerten oder Inhalte der Webseiten verändern. Es erweitert die Möglichkeiten von HTML und CSS um alle Themengebiete, die mit Interaktivität zu tun haben. Heutzutage findet man aber JavaScript auch außerhalb von Webbrowsern, etwa auf Servern oder zur App-Programmierung.



Einfache Taschenrechner-App: entwickelt mit HTML, CSS und JavaScript

## M 1

**Hinweis:** Implementieren ist ein Fachbegriff und wird verwendet, wenn man „programmieren“ oder „Programme entwickeln“ meint.

**Hinweis:** „Formatieren“ steht in der Fachwelt für „gestalten“ bzw. „designen“.

### Welche Vorteile bietet JavaScript?

Da JavaScript standardmäßig in HTML eingebunden werden kann, läuft es quasi auf jedem Betriebssystem (plattformunabhängig). Es sind keine weiteren Installationen notwendig. Es benötigt nur einen Browser und einen Code-Editor, die standardmäßig jedes System mitliefert. Ohne größeren Aufwand können dann kleinere Programme zum Beispiel für die Schulhomepage erstellt werden, die auch auf leistungsschwachen Rechnern problemlos laufen. JavaScript ist eine objektbasierte Sprache. Somit können grundlegende Elemente moderner Programmierung (wie z. B. das Prinzip der Datenkapselung) unterstützt werden. Besonders motivierend ist hierbei die syntaktische Ähnlichkeit zur äußerst populären Sprache *Java*. Es eignet sich daher auch dazu, grundlegende Programmiertechniken zu vermitteln, die dann in anderen Programmiersprachen genauso oder so ähnlich Anwendung finden.

### Aufgabe: Welche Bedeutung nimmt JavaScript in der Technik-Welt ein?

Es gibt verschiedene Ranking-Systeme, die Programmiersprachen hinsichtlich Suchmaschinentreffer, Github-Projekten oder auch Jobangeboten vergleichen.

Informiere dich mithilfe des IEEE-Spectrums über die Top Ten der Programmiersprachen. Vergleiche diese hinsichtlich der Bedeutung von JavaScript zu anderen Programmiersprachen. Nutze dazu folgenden Link: <https://spectrum.ieee.org/the-top-programming-languages-2023>



## M 2

**Hinweis:** Kommentare können verwendet werden, um JavaScript-Code zu erklären und ihn lesbarer zu machen. Einzeilige Kommentare beginnen hierbei mit Doppelschrägstrichen //

**Hinweis:** Redundanz liegt vor, wenn dieselbe Information mehrfach abgespeichert wird. Dies kann bei Programmausführung zu zahlreichen Fehlern führen. Daher sollte dies tunlichst vermieden werden.

## Einbettung von JavaScript in HTML

### Variante 1: Einfügen von JavaScript direkt im HTML-Dokument

JavaScript-Quelltexte können zum einen in HTML in einem `<script>`-Element notiert oder referenziert werden. Das `<script>`-Element darf dabei im `<head>` oder `<body>` des HTML-Dokuments notiert werden.

```
<script>

    // hier kommen JavaScript-Anweisungen;

</script>
```

### Variante 2: JavaScript als externe Datei einbinden

Zum anderen können JavaScript-Quelltexte auch über eine separate Datei (z. B. *skript.js*) eingebunden werden. Hierzu muss dem Skript die Quelladresse der Skriptdatei mittels des Attributbefehls `src=""` im Anfangstag übergeben werden. Um ein reibungsloses Interpretieren des Browsers zu ermöglichen, sollte dazu der `<script>`-Tag im `<head>` stehen.

```
</head>

<script src="skriptname.js"></script>

</head>
```

#### Wie sollte man bevorzugt Skripte einbauen?

Wenn ein Browser eine Webseite lädt, folgt er einer festen Routine. Früher empfahl man deshalb, JavaScript am Schluss einzubinden, indem das `<script>`-Element vor das schließende `<body>`-Tag gesetzt wurde. Dieser Ansatz birgt aber das Problem, dass dadurch der Browser die Skripte erst downloaden kann, wenn das ganze HTML-Dokument geladen und geparkt ist. Gerade bei größeren Seiten mit vielen Stylesheets, Skripten und eingebundenen Bibliotheken kann dies zu einer schlechteren Performance führen, wenn Nutzerinnen und Nutzer nach erfolglosem Warten auf der Webseite entnervt aufgeben. Heute würde man Skripte daher tendenziell im `<head>` einfügen.

Das direkte Notieren aller JavaScript-Anweisungen in HTML ist dennoch im Allgemeinen nicht mehr zu empfehlen. Viel besser ist es, Skripte über externe Dateien einzubinden. Dadurch ist es möglich, JavaScript-Dateien in beliebig viele Webseiten einzubinden und Redundanzen von Quellcode zu vermeiden.

#### Fazit

Skripte gehören in den `<head>`-Bereich einer Webseite – vorzugsweise ausgelagert in eine externe Datei. Durch die Attribute wie `async` können Skripte dem Browser mitteilen, dass dieser mit dem Parsen fortfahren kann, während große Skripte weiter heruntergeladen werden.

```
<script src="skript1.js" async></script>
<script src="skript2.js" async></script>
```

Die wenigen alten Browser, die ein solches Attribut noch nicht kennen, werden es ignorieren. Stattdessen wird unser Skript synchron geladen und sofort ausgeführt. Das wäre auch in Ordnung.

## Funktionen in JavaScript

## M 5

### Lass uns zaubern!

Stell dir vor, du hast einen Zauberstab. Dieser Zauberstab kann spezielle Zaubersprüche ausführen. Jedes Mal, wenn du einen bestimmten Zauberspruch aussprichst, führt der Zauberstab eine bestimmte magische Handlung aus. Du könntest z. B. den Zauberstab mit dem Spruch „Licht!“ dazu bringen, Licht zu erzeugen, oder mit „Fliegen!“ in die Luft zu steigen.

In der JavaScript-Welt sind *Funktionen* wie diese Zaubersprüche. Sie haben einen Namen und führen eine bestimmte Aufgabe aus, wenn sie „aufgerufen“ oder „ausgesprochen“ werden (siehe Arbeitsblatt Events M 6).

### Beispiel:

```
function lichtErzeugen() {  
    console.log(„Das Licht ist jetzt an!“);  
}
```

Rufst du `lichtErzeugen()` auf, wird „Das Licht ist jetzt an!“ in der Konsole angezeigt.

Zu Beginn steht also immer das Schlüsselwort `function`. Es zeigt JavaScript an, dass wir gerade dabei sind, eine neue Funktion zu definieren. Direkt nach dem Schlüsselwort geben wir der Funktion einen *Namen*. Dieser Name ist essenziell, da wir ihn später nutzen, um auf die Funktion zuzugreifen und sie auszuführen.

### Stell dir eine Maschine vor ...

Stell dir nun eine Maschine in einer Fabrik vor. Diese Maschine hat einen Einwurfschlitz, in den du Rohstoffe oder Zutaten legen kannst. Nachdem du die Zutaten eingefügt hast, drückst du den Startknopf und die Maschine verarbeitet diese Zutaten und produziert ein fertiges Produkt.

In JavaScript sind Funktionen ähnlich. Du kannst ihnen „Zutaten“ oder „Rohstoffe“ geben, die wir *Parameter* nennen. Nach dem Funktionsnamen kommen also immer runde Klammern `()`. Innerhalb dieser Klammern können wir diese Parameter definieren. Man kann sich Parameter als die Rohstoffe oder Zutaten vorstellen, die unsere Maschine benötigt, um ihre Arbeit zu verrichten. Es könnte eine, zwei oder auch gar keine Zutat geben – das hängt ganz von der spezifischen Funktion ab.

Die Funktion verarbeitet diese. Dies stellt das Herzstück jeder Funktion dar: dem Anweisungsblock, der von geschweiften Klammern `{ }` umgeben ist. Hier legen wir fest, welche Aufgaben unsere Maschine ausführen soll.

Funktionen geben in der Regel ein Ergebnis, das Produkt, zurück. Innerhalb dieses Blocks kann dann die Anweisung `return` stehen, die uns erlaubt, ein fertiges Produkt oder Ergebnis aus unserer Funktion herauszugeben.

### Beispiel:

```
function macheSaft(frucht) {  
    return frucht + „saft“;  
}
```

Wenn du `macheSaft(„Apfel“)` aufrufst, gibt die Maschine, also die Funktion, „Apfelsaft“ zurück.

Bestandteile einer Funktion in JavaScript im Überblick:

Bestandteil	Beschreibung
Schlüsselwort <code>function</code>	Das Schlüsselwort, das den Beginn einer Funktionsdefinition anzeigt.
Funktionsname	Der eindeutige Name der Funktion, mit dem diese später aufgerufen wird.
Parameter (optional)	Variablen, die in den runden Klammern <code>()</code> definiert werden und Werte repräsentieren, die die Funktion verwendet.
Anweisungsblock	Ein Satz von Anweisungen, eingeschlossen von geschweiften Klammern <code>{ }</code> , der festlegt, was die Funktion tut.
Rückgabewert <code>return</code> (optional)	Ein Schlüsselwort, das innerhalb des Anweisungsblocks verwendet wird, um einen Wert aus der Funktion zurückzugeben.

### Dein Werkzeugkasten

Nicht immer musst du alle Funktionen selbst erfinden. Es gib vorverpackte Werkzeuge und vorgefertigte Lösungen, die uns helfen. In der Programmierung sind das *Bibliotheksfunktionen*. Diese sind so konzipiert, dass sie gängige Aufgaben erledigen, ohne dass wir diese von Grund auf neu erstellen müssen. In JavaScript gibt es viele solcher Funktionen. Hier sind einige Beispiele:

Funktion	Beschreibung	Beispiel
<code>alert()</code>	Zeigt ein Pop-up-Fenster im Webbrowser mit einer Nachricht an. Es stellt ein einfaches Mittel dar, um Benutzer schnell über Informationen oder Warnungen zu informieren.	<pre>alert("Warnung: Es ist ein Fehler aufgetreten.");</pre>
<code>Date()</code>	Ermöglicht das Arbeiten mit Datums- und Zeitangaben.	<pre>let datum = new Date();</pre>
<code>Math.random()</code>	Gibt Zufallswerte an, die zwischen 0 und 1 liegen. Oft wird es in Kombination mit anderen Mathematikfunktionen verwendet	<pre>let zahl = Math.floor(Math.random() * 10) + 1; // liefert eine Zufallszahl zwischen 1 und 10</pre>