JavaScript Grundlagen

Ralph Steyer

1. Ausgabe, Juni 2023

ISBN 978-3-98569-151-7

JAVS_2023

Plus

HERDT

Bevor Sie beginnen …

Voraussetzungen und Ziele

Zielgruppe

Dieses Buch richtet sich an Webseitenersteller und Webdesigner sowie Programmierer, die JavaScript einsetzen möchten. Dabei spielt es keine Rolle, ob diese z. B. bereits professionelle Webseiten für ein Internetportal oder nur für die eigene Homepage erstellt haben. Auch für Umsteiger aus anderen Programmiersprachen ist das Buch geeignet.

Empfohlene Vorkenntnisse

Für das Verstehen dieses Buches werden grundlegende Kenntnisse in HTML vorausgesetzt. Auf die Bedeutung von HTML-Tags wird nicht eingegangen. Erfahrungen in anderen Programmier- oder Skriptsprachen sind nicht notwendig. Der Umgang mit dem eigenen Betriebssystem und typischen Programmen wie einem Editor oder Browser wird vorausgesetzt.

Hinweise zur Software

- JavaScript ist nicht von der Verwendung eines bestimmten Betriebssystems abhängig. Sie können Linux, macOS, Windows oder auch ein anderes Betriebssystem verwenden, wenn dafür JavaScript-Unterstützung angeboten wird. Es wird aber davon ausgegangen, dass das Betriebssystem nicht zu alt ist.
- Sofern nicht anders vermerkt, wird im Folgenden weiter davon ausgegangen, dass Sie einen oder mehrere der unten genannten Browser in aktuellen Versionen verwenden. Beachten Sie, dass die meisten Browser mittlerweile einen äußerst kurzen Versionszyklus einhalten und in einem kurzen Zeitraum mehrere Versionen erscheinen, die sich oft nur geringfügig unterscheiden. Ältere Browser werden in dem Buch nicht mehr berücksichtigt.

Name	Erhältlich unter der Internetadresse:
Edge oder dessen Vorgänger Internet Explorer 11	*https://www.microsoft.com/en-us/edge*
Mozilla Firefox	*https://mozilla-firefox.de.uptodown.com*
Opera	*https://www.opera.com/de*
Apple Safari	*https://www.apple.com/de/safari/*
Google Chrome	*https://www.google.com/chrome/*

Sofern Sie serverseitiges JavaScript bzw. JavaScript unabhängig von einem Browser nutzen wollen, ist die Installation von **Node.js** zu empfehlen. Sie finden die neuste Version von Node.js unter *https://nodejs.org/*. Sie erhalten dort einen Installationsassistenten für Ihr Betriebssystem, der Ihnen die einfache Installation des Systems erlaubt.

Node.js ist eine plattformübergreifende Open-Source-JavaScript-Laufzeitumgebung (JavaScript-Engine). Damit kann JavaScript-Code unabhängig von einem Webbrowser ausgeführt und etwa JavaScript bei einen Webserver verwendet werden. Node.js wird in der JavaScript-Laufzeit-umgebung V8 ausgeführt, die ursprünglich für Google Chrome entwickelt wurde.

Die Verwendung von JavaScript in Node.js entspricht damit im Wesentlichen der Verwendung in der Konsole eines Browsers. Man ist losgelöst von der HTML-Umgebung und dem DOM einer Webseite und reduziert auf pures JavaScript. Allerdings kann diese Umgebung durch diverse Module und Objekte erweitert werden, was die Einsatzmöglichkeiten von JavaScript auf einen Umfang „normaler“ Programmiersprachen erweitert.

Konventionen

Hervorhebungen im Text

Im Text erkennen Sie bestimmte Programmelemente an der Formatierung. So werden Datei- und Ordnernamen, Hyperlinks und Bezeichnungen für Menüs bzw. Menüpunkte und Programmelemente wie Register oder Schaltflächen immer *kursiv* geschrieben und wichtige Begriffe **fett** hervorgehoben.

`Courier New`	kennzeichnet Programmtext, Programmnamen, Funktionsnamen, Variablennamen, Datentypen, Operatoren etc.
`Courier New kursiv`	kennzeichnet Zeichenfolgen, die vom Anwendungsprogramm ausgegeben oder in das Programm eingegeben werden.
[ ]	Bei Darstellungen der Syntax einer Programmiersprache kennzeichnen eckige Klammern optionale Angaben.
/	Bei Darstellungen der Syntax einer Programmiersprache werden alternative Elemente durch einen Schrägstrich voneinander getrennt.

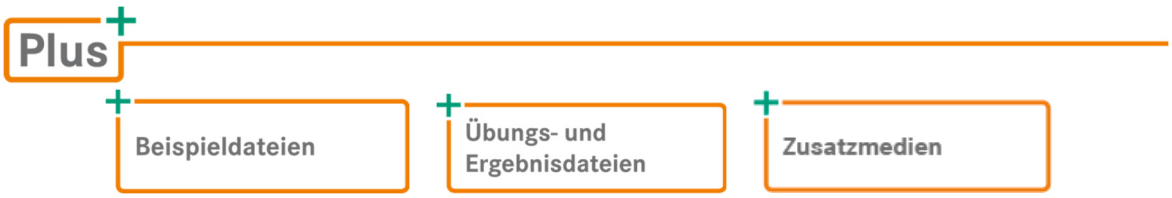
In den abgebildeten Beispielen wird aufgrund der verwendeten Dokumenttyp-Deklaration auf die exakte Schreibweise der öffnenden und schließenden Tags verzichtet. Darum werden z. B. `<p>`-Tags ohne das schließende `</p>`-Tag verwendet.

HERDT BuchPlus – unser Konzept:

Problemlos einsteigen – Effizient lernen – Zielgerichtet nachschlagen

(weitere Infos unter *www.herdt.com/BuchPlus*)

Nutzen Sie dabei unsere maßgeschneiderten, im Internet frei verfügbaren Medien:



Wie Sie schnell auf diese BuchPlus-Medien zugreifen können, erfahren Sie unter *www.herdt.com/BuchPlus*.

In den Übungsdateien wird teils über die Meta-Angabe `charset` die Zeichencodierung UTF-8 verwendet. Damit ist es nicht mehr notwendig, dass Sie z. B. für deutsche Umlaute die Zeichenreferenz angeben. Dazu wurden die Übungsdateien im Unicode-Format abgespeichert und müssen daher auch in einem Unicode-fähigen Texteditor geöffnet werden, um sie verändern zu können.

1

Einführung in JavaScript

1.1 Entstehung von JavaScript

Die Sprache JavaScript wurde 1995 von der Firma Netscape Communications in Kooperation mit der Firma Sun Microsystems ins Leben gerufen. Der ursprüngliche Name war LiveScript. Aus Marketinggründen wurde ein ähnlicher Name wie für das Sun-Produkt Java gewählt, um eine gemeinsame Vermarktungsstrategie zu gewährleisten. Aufgrund der Ähnlichkeit der Namen JavaScript und Java kommt es häufig zu Verwechslungen. Die Gemeinsamkeiten zwischen den Sprachen sind zwar syntaktisch groß, aber vom Konzept gibt es starke Abweichungen. Eingeführt wurde JavaScript im damaligen Browser Netscape Navigator 2.

Im Zuge der Standardisierung von JavaScript arbeitete die Firma Netscape später mit der ECMA (**E**uropean **C**omputer **M**anufacturers **A**ssociation) zusammen. Die daraus resultierende Spezifikation wurde ECMA-262 mit weiteren Versionen (Editionen) genannt. Die offizielle Version von JavaScript heißt deshalb ECMAScript.

Nach dem Ende der Firma Netscape Communications und der Einstellung des Browsers Netscape Navigator übernahmen die Mozilla Foundation (Browser Firefox) und ECMA die Weiterentwicklung der Skriptsprache.

1.2 Grundlagen zu JavaScript

Wie Sie JavaScript einsetzen können

JavaScript wurde entwickelt, um die statischen Inhalte von Webseiten mit dynamischen Inhalten und Interaktionen zu erweitern. Dies sollte letztendlich dazu dienen, Webseiten attraktiver zu gestalten und den Nutzern eine verbesserte Funktionalität zu bieten. Heutzutage erlangt JavaScript in Zusammenhang mit HTML5 und CSS3 und den modernen **Rich Internet Applications** (RIAs) immer mehr an Bedeutung beim Erstellen von applikationsähnlichen Programmen. Ebenso wird JavaScript immer mehr auf Serverseite bzw. allgemein unabhängig von der Integration in einem Browser eingesetzt. JavaScript wird z. B. für die folgenden Anwendungsfälle genutzt:

- Bevor Sie die Inhalte von Formularen abschicken, können Sie über JavaScript die korrekte Eingabe der Daten sicherstellen. So können Sie beispielsweise prüfen, ob eine Postleitzahl nur aus Zahlen besteht. Diese Prüfung erfolgt ohne Kommunikation mit dem Webserver. Dieser wird dadurch entlastet, und der Benutzer wird schneller auf Fehler hingewiesen.
- Sie können vollständige Anwendungen entwickeln, die in einer Webseite direkt beim Nutzer ausgeführt werden. Das können kleine Anwendungen wie z. B. ein Taschenrechner oder Währungsumrechner sein. Aber auch größere Anwendungen wie interaktive Kalender, Portale etc. sind mit JavaScript möglich – insbesondere, wenn diese mit dem Webserver im Hintergrund kommunizieren.
- Es ist die Darstellung von dynamischen Inhalten möglich, ohne dass dazu Skripts auf dem Webserver oder spezielle Erweiterungen in dem Browser notwendig sind. So können Sie beispielsweise nach der Eingabe von Messdaten ein Diagramm anzeigen oder je nach eingegebenen Werten dynamisch weitere Formularelemente anzeigen bzw. verbergen.
- Sie können Daten zwischen Browser und Webserver austauschen, ohne dass Sie die Webseite im Browser neu laden müssen. Ein bekannter Einsatz ist hierbei die Anzeige von Suchergebnissen, während Sie noch die Suchanfrage eingeben.
- Sie können Daten auf dem Client speichern.
- Sie können den Ort des Besuchers einer Webseite bestimmen und ihm damit lokalisierte Informationen anzeigen.
- Sie können zahlreiche Daten des Clientrechners auswerten und damit Seiten und Inhalte an die Gegebenheiten beim Besucher anpassen.
- Auf einem Server, der JavaScript unterstützt, können Sie alle Aktionen durchführen, die man etwa mit speziell ausgerichteten Sprachen und Techniken wie PHP, Perl, Java Server Pages, ASP.NET etc. macht, beispielsweise Daten vom Client entgegennehmen und verwerten oder auf serverseitige Datenbanken zugreifen.
- JavaScript kann bei Desktop-Applikationen, deren Oberfläche mit einer XML-Struktur beschrieben wird (etwa FXML bei JavaFX), direkt zur Programmierung von Ereignisroutinen eingesetzt werden.

Was ist JavaScript?

JavaScript ist eine Skriptsprache, mit der Sie vollwertig programmieren können, im Gegensatz zur Auszeichnungssprache HTML. Da JavaScript mit Objekten arbeitet, wird auch von einer objektbasierten oder (früher mit Abstrichen) objektorientierten Sprache gesprochen. Da ebenso funktionales sowie prozedurales Programmieren möglich ist, kann JavaScript sehr flexibel eingesetzt werden.

Die Verwendung von JavaScript ist im Gegensatz zu vielen anderen Programmiersprachen recht einfach zu erlernen, da sie weniger Sprachelemente besitzt und viele Dinge automatisch erledigt. Dennoch nimmt die Einarbeitung einige Zeit in Anspruch, da Sie nicht nur die Sprache, sondern auch das Einsatzgebiet kennenlernen müssen. Gerade die Einschätzung der sehr unterschiedlichen Verhaltensweisen von Browsern erforderte in der Vergangenheit viel Erfahrung, und bei richtig komplexen Applikationen können auch JavaScripts sehr umfangreich werden. In der letzten Zeit wird JavaScript auch immer professioneller in großen Projekten eingesetzt und löst zum Teil mächtige Konkurrenztechnologien wie Java oder C# in geeigneten Umgebungen ab. Viele Programmierer mit Erfahrung in C#, Java oder C/C++ eignen sich deshalb JavaScript-Kenntnisse an.

JavaScript einsetzen

JavaScript kann im Browser (clientseitig – beim Benutzer) und in einigen Webservern (serverseitig) sowie in einer allgemeinen JavaScript-Engine eingesetzt werden. Benutzer einer Webseite bekommen die Art des Einsatzes von JavaScript auf einem Webserver nicht mit, da in der Webseite nur Kontakt mit clientseitigem JavaScript besteht und der Webserver eine „Blackbox" darstellt. JavaScript wird immer noch überwiegend in Browsern genutzt, gewinnt aber jenseits dieses Einsatzzwecks aktuell rasant an Bedeutung.

Die Implementierung von JavaScript erfolgt clientseitig direkt im Browser, indem ein Skript in eine Webseite eingebunden wird. JavaScript-Anwendungen können clientseitig nicht außerhalb, sondern nur innerhalb eines Browsers bzw. einer entsprechenden Ausführungsumgebung ausgeführt werden. Diese Abschirmung vom Rest des Computersystems im Browser wird auch als Sandbox bezeichnet.

Da Browser unter verschiedenen Betriebssystemen zum Einsatz kommen, wird JavaScript auch als plattformübergreifende Sprache bezeichnet. Der Vorteil für Sie ist dabei, dass ein JavaScript-Programm theoretisch unter verschiedenen Browsern und Betriebssystemen gleichermaßen läuft. Dies war lange Zeit aufgrund der unterschiedlichen JavaScript-Versionen und Implementierungen im Browser oft nur für einfache Skripts der Fall, aber mittlerweile geht das auch recht zuverlässig für komplexere Skripts.

Wie wird JavaScript in eine Webseite eingebunden?

Mit JavaScript können Sie unter anderem auf HTML-Elemente zugreifen bzw. HTML-Code erzeugen. Der JavaScript-Code wird dazu in das HTML-Dokument eingefügt oder über eine separate Datei eingebunden. Da sich der JavaScript-Code in einem für Menschen lesbaren Zustand befindet, muss er vom Browser erst interpretiert werden (Prüfung der syntaktischen Korrektheit, Übersetzung, Ausführung).

JavaScript wird in den Quelltext eines HTML-Dokuments eingebunden und gesondert gekennzeichnet. Die Einbindung erfolgt über das Tag `<script>`, das mit einem Ende-Tag abgeschlossen werden muss. Bei der Darstellung des HTML-Dokuments kann der Browser die relevanten Stellen zwischen den beiden `<script>`-Tags als JavaScript identifizieren und sie entsprechend verarbeiten. Browser, die kein JavaScript verstehen oder bei denen JavaScript deaktiviert ist, ignorieren diese Anweisungen.

Browser, die kein JavaScript verstehen, wurden meist vor 1995 entwickelt und werden nicht mehr verwendet.

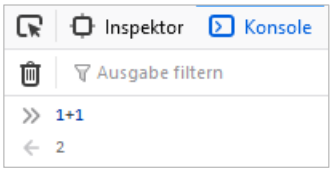
```
<script type="text/javascript">
  <!-- JavaScript-Anweisungen -->
</script>
```

Das Attribut `type` kann entfallen, denn Browser werden als Vorgabe immer JavaScript verwenden. Sie geben mit dem Attribut bei einer vollständigen Notation allgemein den sogenannten MIME-Type (**M**ultipurpose **I**nternet **M**ail **E**xtensions) an. Mit HTML5 wird das Attribut offiziell nicht mehr gefordert. Die Beibehaltung des Attributs ist aber empfehlenswert, um deutlich zu kennzeichnen, dass man mit JavaScript arbeitet.

Wie nutzt man JavaScript ohne Integration in eine Webseite?

Traditionell ist die Verwendung von JavaScript mit einer Integration in eine Webseite verbunden. Aber diese Beschränkung ist schon lange gefallen. JavaScript kann entweder in der Konsole eines Webbrowsers direkt ausgeführt werden oder man verwendet eine JavaScript-Laufzeitumgebung wie Node.js. In beiden Fällen wird keine Webseite als Gerüst bzw. Verankerung des Skripts mehr benötigt. Allerdings können Sie dann auch nicht auf die Bestandteile der Webseite wie Formulare oder Grafiken zugreifen. Auch nutzen einige auf XML basierende Oberflächentechnologien JavaScript zur Programmierung von dynamischen Verhaltensweisen.

Wenn Sie in einem Browser die Konsole öffnen (beispielsweise mit einem Rechtsklick in eine Webseite und Öffnen der Entwicklertools unter Firefox – bei anderen Browsern geht das ähnlich), können Sie dort direkt JavaScript-Anweisungen notieren, z. B. die Berechnung des Ergebnisses von `1 + 1`. Das Ergebnis wird Ihnen direkt in der Konsole angezeigt.



Wenn Sie **Node.js** mit Standardeinstellungen installiert haben, öffnen Sie eine Eingabeaufforderung/Shell/Konsole Ihres Betriebssystems und geben dort *node* ein. Sie gelangen danach in den Eingabemodus von Node.js und können dort, vollkommen analog wie in der Browserkonsole, JavaScript-Befehle eingeben.

```
C:\Users\ralph>node
Welcome to Node.js v18.14.0.
Type ".help" for more information.
> 1+1
2
>
```

Mit der Anweisung `.quit` (beachten Sie den vorangehenden Punkt) beenden Sie die Node.js-Konsole.

Eine JavaScript-Datei können Sie auch als Parameter an Node.js übergeben und damit direkt ausführen, ohne in den Eingabemodus von Node.js gehen zu müssen.

Etwa führen Sie so die JavaScript-Datei *javascriptdatei.js* aus, wenn Sie sich in dem Verzeichnis mit der JavaScript-Datei befinden:

node javascriptdatei.js

Sicherheit

Das Ausführen von JavaScript-Code kann vom Anwender in den Einstellungen des Browsers deaktiviert werden. Früher haben einige Benutzer JavaScript auch wegen Sicherheitslücken durch fehlerhafte Implementierungen in den Browsern ausgeschaltet. Die Gefahr durch JavaScript-Anwendungen ist jedoch gering, denn diese laufen im Browser in einer abgeriegelten Umgebung ab, in der schon erwähnten Sandbox. Sie können nicht auf Dateien des lokalen Rechners zugreifen (Ausnahme – dafür explizit vorgesehene Dateien wie der lokale Speicher) und auch keine Benutzerdaten abfragen, die nicht auch über den Browser übermittelt werden können.

Zudem werden die JavaScript-Implementierungen in den aktuellen Browsern immer besser und sicherer. Mittlerweile haben fast alle Anwender JavaScript aktiviert, denn moderne Webseiten sind ohne JavaScript meist nicht mehr zu verwenden, und Browser haben JavaScript in der Standardeinstellung aktiviert. Die Browserhersteller verstecken zudem die Möglichkeiten zur Deaktivierung von JavaScript so, dass typische Anwender sie gar nicht finden.

Wenn Sie JavaScript außerhalb des Browsers ausführen, steht dort die Sandbox des Browsers **nicht** zur Verfügung. Was dann mit JavaScript möglich ist, unterliegt der JavaScript-Engine. Allerdings eröffnet dies viel weitergehende Möglichkeiten, die Sie dann mit JavaScript haben.

1.3 JavaScript-Versionen

Die Sprache JavaScript wird von der Mozilla Foundation immer wieder erweitert, wobei die **praxisrelevante** Entwicklung im Grunde seit dem Jahr 2000 stagniert hat. Die Neuerungen aller späteren Vorschläge der Mozilla Foundation wurden gar nicht oder nicht einheitlich von den unterschiedlichen Browsern interpretiert, mit wenigen Ausnahmen, die aber nicht explizit zu neuen Versionsnummern von JavaScript geführt haben.

Die Weiterentwicklung von JavaScript wird allerdings durch HTML5 vorangetrieben, und viele der proprietären JavaScript-Neuerungen der Mozilla Foundation, die seit dem Jahr 2000 für Firefox und seine Verwandten eingeführt wurden, halten über diesen „Marketingumweg“ Einzug in andere moderne Webbrowser. Auch haben populäre JavaScript-Erweiterungen wie Ajax jenseits der „offiziellen“ JavaScript-Versionen immer wieder für eine Weiterentwicklung der JavaScript-Features gesorgt.

Beachten Sie, dass Microsoft kein JavaScript, sondern eine eigene Skriptsprache mit dem Namen **JScript** verwendet. Diese ist zwar fast identisch zu JavaScript, denn sie basiert wie JavaScript selbst auf der ECMA-Spezifikation. Allerdings enthält JScript einige Erweiterungen und kleinere Abweichungen von JavaScript, was in der Vergangenheit in gewissen Situationen eine besondere Programmierung für den Internet Explorer notwendig machte. Diese Ausnahmesituationen nehmen stark ab, da der Internet Explorer mehr oder weniger Geschichte ist und sich dessen Nachfolger Edge weitgehend standardkonform verhält.

Weiter gibt es noch **TypeScript**. Dies ist eine Open-Source-Programmiersprache, die von Microsoft entwickelt und gepflegt wird. Es ist ein Superset von JavaScript und fügt der Sprache optional sogenannte statische Typisierung (und früher klassenbasiertes objektorientiertes Verhalten – das kann neues JavaScript aber auch) hinzu. In Browsern wird TypeScript aber – sofern unterstützt – in normales JavaScript umgewandelt, bevor die Ausführung erfolgt.

Die jeweiligen Versionen von JavaScript werden von der Mozilla Foundation mit einer fortlaufenden Nummerierung gekennzeichnet und starteten im März 1996 mit der Version 1.0. Die derzeit letzte offizielle Versionsnummer 1.8.5 stammt vom Juli 2010 und ist mit ECMAScript 1.8.1 kompatibel. Tatsächlich wurden die offiziellen Versionsnummern von JavaScript damit quasi „eingefroren“, aber die zugrundliegende Basis ECMAScript weiter vorangetrieben.

Die zum Zeitpunkt der Bucherstellung aktuellste Version ECMAScript 2022 stammt vom Juni 2022 und wird von den meisten Browsern nicht vollständig unterstützt. Aber das soll keinesfalls bedeuten, dass sich im Umfeld von JavaScript nichts tut.

Vereinheitlichungen in Browsern, Optimierungen der Ausführungsumgebungen und vor allen Dingen die Unterstützung zusätzlicher Features und Definitionen aus modernen Versionen von ECMAScript erfolgen permanent. Die meisten Neuerungen sind allerdings für den Einstieg in JavaScript sowieso von geringer Bedeutung.

JavaScript-Versionen angeben

In modernen Webseiten gibt man normalerweise die verwendete JavaScript-Version nicht mehr an, da alle modernen Browser in etwa die gleichen Features unterstützen. Im Prinzip kann man bei der Einbindung von JavaScript jedoch eine bestimmte JavaScript-Version angeben. Damit können Sie inkompatible Browser von dem verwendeten Skript-Bereich fernhalten. Dazu können Sie die jeweilige Version mit dem Attribut `type` im `<script>`-Tag angeben. Unterstützt ein Browser nur eine niedrigere Versionsnummer, soll der entsprechende JavaScript-Code nicht ausgeführt werden. Andernfalls würden die neuen und damit unbekannten JavaScript-Befehle einen Fehler produzieren. Aber diese Vorsicht ist aus genannten Gründen in der Praxis mittlerweile unbegründet.

```
<script type="text/javascript; version=1.6">
  <!-- JavaScript-Anweisungen -->
</script>
```

Diese Versionsangabe im `<script>`-Tag über einen speziellen Wert im Attribut `type` ist nicht standardisiert und wird zurzeit nur von Mozilla Firefox und dessen Verwandten ausgewertet. Früher hat man bei der Überprüfung der maximal möglichen JavaScript-Version stattdessen das eigentlich veraltete `language`-Attribut verwendet. Die gleichzeitige Verwendung von `type` und `language` muss unterbleiben, da `language` immer ignoriert wird, wenn Sie gleichzeitig auch `type` notieren. Aber die Angabe der Version sollte wie gesagt unterbleiben, wenn es nicht einen unabdingbaren Grund dafür gibt. Im Zweifelsfall handelt man sich damit eher Probleme ein, wenn Browser unnötig ausgegrenzt werden. Denn obwohl moderne Browser viele Features unterstützen, die nach JavaScript 1.5 eingeführt wurden, würde eine Versionsangabe jenseits von JavaScript 1.5 unnötigerweise verhindern, dass die damit referenzierten Skripte ausgeführt werden.

1.4 JavaScript-Aktivierung im Browser testen

Sie können davon ausgehen, dass JavaScript grundsätzlich bei den Anwendern aktiviert ist (Voreinstellung aller Browser) und die meisten Anwender JavaScript nicht deaktivieren. Dennoch können Sie nicht zu 100 % sicher sein, dass JavaScript aktiviert ist. Verwenden Sie das Skript im folgenden Beispiel, um die Aktivierung von JavaScript zu prüfen. Erstellen Sie ein HTML-Dokument, das eine erste einfache JavaScript-Anweisung ausführt. Die Erläuterung der genauen Funktionsweise ist in diesem Moment nicht von Interesse. Sie erfolgt im weiteren Verlauf des Buches.

Beispiel: *test.html*

Geben Sie den folgenden Quellcode in einem Texteditor ein, und speichern Sie ihn in der Datei *test.html*. Nach dem Öffnen des HTML-Dokuments in Ihrem Browser sollte sich ein Popup-Fenster öffnen.

```
<!DOCTYPE html>
① <html>
    <head>
      <meta charset="utf-8">
      <title>Der erste JavaScript-Test</title>
    </head>
    <body>
②     <script type="text/javascript">
③       alert('Glückwunsch! JavaScript ist in Ihrem Browser
              aktiviert.');
④     </script>
    </body>
</html>
```

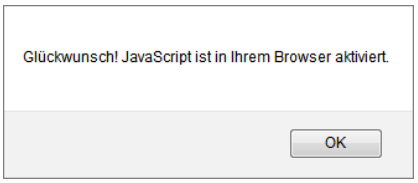
① Das Grundgerüst eines HTML-Dokuments wird eingeleitet.

② Dem Browser wird über die Anweisung `<script type="text/javascript">` mitgeteilt, dass die Anweisungen der verwendeten Skriptsprache als JavaScript zu interpretieren sind.

③ Die JavaScript-Funktion `alert()` zeigt ein Meldungsfenster an. Als Text wird die angegebene Zeichenkette eingeblendet.

Wird kein Fenster eingeblendet, ist JavaScript nicht aktiviert, sofern Sie keinen Fehler gemacht haben.

④ Mit dem schließenden Tag `</script>` wird der JavaScript-Teil beendet.



JavaScript-Meldung im Mozilla Firefox

1.5 Nützliche Webseiten

https://developer.mozilla.org/en/Javascript	Informationen zu JavaScript
https://wiki.selfhtml.org/wiki/JavaScript	Einführung in JavaScript (SelfHTML, deutsch)
https://www.jswelt.de/	Umfangreiche JavaScript-Sammlung
https://nodejs.org/de	Deutsche Seite von Node.js

Plus **Ergänzende Lerninhalte:** *Informationen im Internet.pdf*

1.6 Übung

Theoriefragen zu JavaScript

Übungsdatei: -- **Ergebnisdatei:** *kap01/uebung.pdf*

1. Wo werden JavaScript-Anweisungen ausgeführt?
2. Erklären Sie die Unterschiede der Sprachen TypeScript, JScript, Java und JavaScript.
3. Zählen Sie einige der möglichen Einsatzgebiete von JavaScript auf.

2

Grundlegende Sprachelemente

2.1 JavaScript in HTML verwenden

JavaScript direkt einbinden

Über das `<script>`-Tag können Sie JavaScript-Code in ein HTML-Dokument einfügen. Der JavaScript-Bereich wird durch Angabe des `</script>`-Tags wieder beendet.

```
<script type="text/javascript">
  <!-- Hier werden die JavaScript-Anweisungen eingefügt -->
</script>
```

Der JavaScript-Code kann sowohl im `head`-Bereich (engl.: head = Kopf) als auch im `body`-Bereich (engl.: body = Körper, Rumpf) einer HTML-Datei eingefügt werden. Beim Laden eines HTML-Dokuments werden die JavaScript-Anweisungen sequenziell abgearbeitet. In der Regel werden Skripts im `<head>`-Tag eingefügt, um oft verwendete Funktionen und globale Variablen anzulegen (zu deklarieren). Skripts im `<body>`-Tag werden z. B. dazu genutzt, eine Ausgabe in das HTML-Dokument durchzuführen.

Innerhalb des `<script>`-Tags wird über das Attribut `type` der MIME-Typ der Skriptsprache angegeben, wobei diese Angabe mit HTML5 nicht mehr gefordert wird.

- Durch die Angabe des optionalen Attributs `defer` (ohne Wert) im `<script>`-Tag teilen Sie dem Browser mit, dass der JavaScript-Code keine Ausgabe im HTML-Dokument erzeugt. Diese Angabe ist rein informell und dient dazu, dass der Browser dadurch das HTML-Dokument schneller darstellen kann, da der JavaScript-Code nicht sofort analysiert (geparst) werden muss, z. B. `<script type="..." defer>`.
- Bei Bedarf können Sie im `<script>`-Tag das Attribut `lang` setzen, um eine Sprache anzugeben, die in dem Quellcode verwendet wird. Beispielsweise `<script lang="en" ...>`, wenn Sie Englisch in dem Quellcode verwenden.
- Wenn Sie über JavaScript Ausgaben im HTML-Dokument vornehmen, wird dieser Code erneut interpretiert und die Webseite neu aufgebaut. Auf diese Weise können Sie über JavaScript beispielsweise JavaScript-Code erzeugen, der ebenfalls interpretiert und ausgeführt wird. Für das Erzeugen umfangreicher dynamischer Webseiten ist dies eine gängige Praxis.

In einer Webseite kann es mehrere Skriptbereiche geben. Diese bilden einen gemeinsamen **Namensraum**. Namensraum (engl. namespace) bedeutet, dass Elemente, die Sie in einem Skriptbereich deklariert haben, auch in den anderen (nachfolgenden) Skriptbereichen verfügbar sind. Ebenso müssen Bezeichner in einem Namensraum eindeutig sein. Das gilt unabhängig von der Art der Einbindung.

Einbinden als externe Dateien

JavaScript-Code können Sie auch als separate Dateien einbinden. Dies ist z. B. nützlich, wenn Sie Funktionen in mehreren HTML-Dokumenten verwenden möchten. Durch die Deklaration der Funktionen in einer Datei steht Ihnen eine zentrale Stelle zur Verfügung, an der Sie Quellcode anpassen können, ohne alle entsprechenden HTML-Dokumente bearbeiten zu müssen.

```
<script type="text/javascript" src="URL"></script>
```

Eigene JavaScript-Dateien werden meist in einem separaten Unterverzeichnis gespeichert. In der Praxis haben sich Verzeichnisstrukturen wie *lib/js* (*lib* steht für Library-Bibliothek), *lib/scripts* oder nur *scripts* oder *js* etabliert.

Sie finden alle JavaScript-Dateien in den BuchPlus-Dateien im Unterverzeichnis *lib/js* der Kapitel.

Die (ausschließliche) Verwendung externer JavaScript-Dateien ist bei modernen Webseiten der Regelfall. Nur damit können Sie die Trennung von Struktur und Funktionalität erreichen, was bei komplexeren Webseiten unabdingbar ist. Beachten Sie, dass in den Beispielen im Buch hin und wieder **rein aus didaktischen Gründen** mit internen Skript-Containern gearbeitet wird. Damit lassen sich Zusammenhänge leichter erklären. In der Praxis können Sie diese internen Skript-Container jederzeit in externe Skripte auslagern.

Über das Attribut `src` beim `<script>`-Tag geben Sie die URL zur Datei an, in der sich der JavaScript-Code befindet. Wie bei der Definition von Hyperlinks können Sie relative und absolute Pfadangaben verwenden. JavaScript-Dateien besitzen standardmäßig die Endung **.js*, wobei das aber nicht zwingend ist.

Eine JavaScript-Datei kann im `head`-Bereich oder `body`-Bereich eines HTML-Dokuments eingebunden werden. Vorzugsweise sollten Sie diese jedoch im `head`-Bereich einbinden, wenn in der JavaScript-Datei **Deklarationen** (das bedeutet die Einführung von neuen Elementen) durchgeführt werden. Erzeugen Sie jedoch direkt Ausgaben in der JavaScript-Datei, sollte die Datei in den `body`-Bereich eingebunden werden, da Sie im `head`-Bereich einer Webseite keine Ausgaben erzeugen können. Genaugenommen wird der Browser diese Ausgaben automatisch aus dem Header der Webseite in den Körper verlagern.

Beispiel: *javascript.js*

In der Datei befindet sich lediglich eine Ausgabeanweisung, die beim Laden der Webseite bzw. externen JavaScript-Datei einen Text in ein HTML-Dokument schreibt.

```
document.write('Ich befinde mich in einer externen Datei.');
```