Marco Emrich und Christin Marit

JavaScript Grundlagen, Sprachkern, Syntax

Ein Webmasters Press Lernbuch

Version 1.10.10 vom 10.02.2022

Autorisiertes Curriculum für das Webmasters Europe Ausbildungs- und Zertifizierungsprogramm

Inhaltsverzeichnis

Vorwort

JavaScript! JavaScript! JavaScript!

Immer häufiger ist diese Sprache anzutreffen. Webentwickler sind sich einig: An JavaScript führt kein Weg vorbei. Das bestätigen beispielsweise auch die *RedMonk Programming Language Rankings* — eine Seite, die Programmiersprachen nach ihrer Popularität bewertet. Dort hält sich JavaScript momentan auf Platz eins, gefolgt von Java, PHP und Python.

Auch wenn diese Class für Anfänger ist, und wir uns bemühen, Ihnen JavaScript möglichst schonend beizubringen, so legen wir doch auch großen Wert auf gute Codequalität. Denn *noch wichtiger als lauffähiger Code ist gut geschriebener Code* — Code, der leicht zu verstehen, leicht zu ändern und leicht zu erweitern ist. Das sagt schon das berühmte *Software Craftsmanship Manifesto*, das Tausende von Softwareentwicklern weltweit unterzeichnet haben.

Zusammengefasst lässt sich sagen: Unser Ziel ist es, Sie in möglichst kurzer Zeit an einen Punkt zu bringen, an dem Sie sauberen, aktuellen JavaScript-Code programmieren.

Sie haben evtl. schon Vorkenntnisse? Dann lassen Sie sich überraschen, und lernen Sie einen neuen Programmierstil kennen, der von den neusten Sprachfeatures profitiert und zu kürzerem, prägnanterem Code führt.

Beim Durcharbeiten, könnte sich Ihnen der Gedanke aufdrängen, dass wir viel Spaß beim Schreiben hatten. Da könnte etwas dran sein :) Wir hoffen, Sie haben nun mindestens genauso viel Spaß beim Lesen und Programmieren…

In diesem Sinne: Ran an den Code!

Ihre Autoren — Christin & Marco

PS: Wenn Sie Feedback und Anregungen haben, oder einfach verfolgen möchten, woran ich gerade arbeite, folgen oder schreiben Sie mir auf Twitter: **@marcoemrich** — Marco

1 Was Sie schon immer über JavaScript wissen wollten…

In dieser Lektion lernen Sie

- was JavaScript ist.
- wo JavaScript eingesetzt wird.
- was Sie in dieser Class erwartet.

Selbst ein Weg von tausend Meilen beginnt mit einem Schritt.

aus Japan

1.1 JavaScript?

Sie haben sich entschlossen, eine Class über JavaScript zu belegen. Vermutlich haben Sie schon die eine oder andere Vorstellung, was Sie mit JavaScript anfangen können. Vielleicht gelingt es uns dennoch, Sie ein wenig damit zu beeindrucken, wie vielseitig und universell die Einsatzmöglichkeiten von JavaScript tatsächlich sind.

JavaScript ist überall!

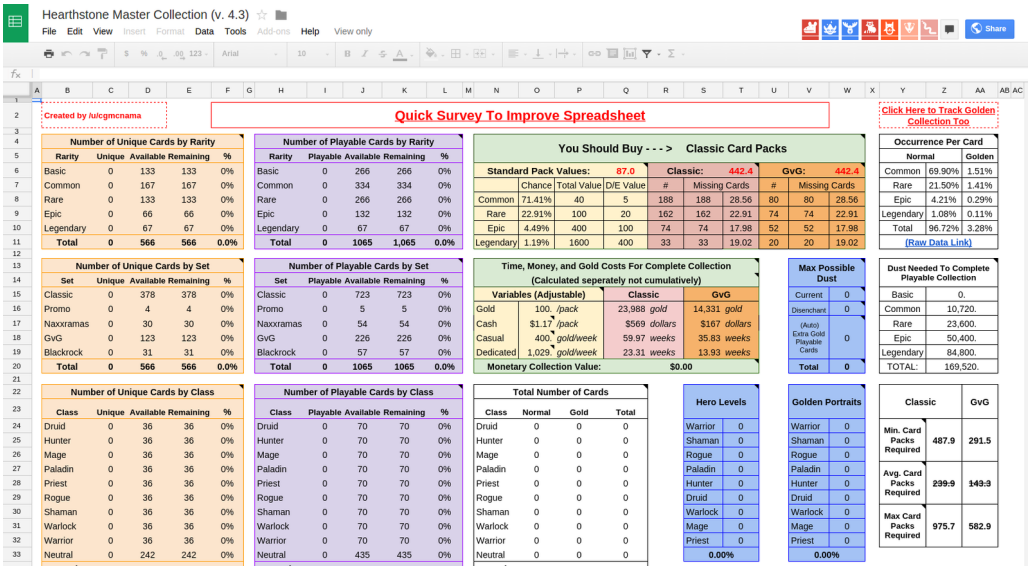


Abb. 1 *Ein Google-Spreadsheet*

JavaScript ist vor allem in Browsern präsent. Mit JavaScript können Sie Programme schreiben, die im Browser ablaufen. Dem Browser kommt dabei die Rolle einer Plattform zu — wie bei einem Betriebssystem, auf dem Sie Programme installieren können. Eine Installation ist aber für JavaScript-Anwendungen nicht nötig.

Die Funktionalität ist dabei sehr weitreichend. Sie fängt an bei einfachen Validierungen von Formularfeldern (z.B. ob eine E-Mail-Adresse richtig geschrieben ist) und reicht bis zu komplexen Anwendungen wie z.B. Google Sheets[1] (eine Art Tabellenkalkulation à la Microsoft Excel im Browser, Abb. 1).

Auch moderne E-Shops (z.B. Amazon, eBay), Videoplattformen (z.B. YouTube) oder Social Sites (Facebook, Twitter) setzen massiv JavaScript ein. Ein modernes Internet ist ohne JavaScript nicht vorstellbar.

Browser können neben JavaScript auch andere Sprachen ausführen, z.B. *Dart* (in Google Chrome) oder *VBScript* (im Microsoft Internet Explorer). JavaScript ist aber die einzige Programmiersprache, die alle modernen Browser zuverlässig unterstützen. Zudem lassen sich enorme Ladeverzögerungen, wie Sie sie vielleicht noch von Java-Applets oder Flash kennen, meistens vermeiden.

Browser sind auch längst nicht die einzige Plattform, die JavaScript ausführen können. So finden Sie JavaScript beispielsweise auch innerhalb von PDFs, zum Scripten von Anwendungen wie Adobe Photoshop oder in der Steuerungssoftware von Robotern[2] (Abb. 2).

Sie sehen: Was Sie in dieser Class lernen, können Sie praktisch überall einsetzen. Dennoch wollen wir uns natürlich vorerst auf den prominentesten Einsatzzweck von JavaScript konzentrieren — den Einsatz im Web.

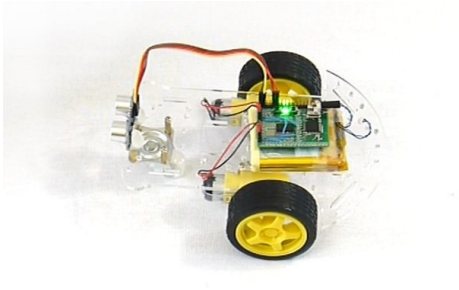


Abb. 2 *Roboter mit Espruino-Board, Foto: Pur3 Ltd*

JavaScript vs. Java

Eine recht hohe Verwechslungsgefahr besteht mit dem Begriff ***Java***. *Java* ist auch eine Programmiersprache, hat aber mit JavaScript so wenig gemein wie *car* mit *carpet*. Netscape hat den vorher verwendeten Namen *LiveScript* abgelöst durch den Namen *JavaScript*, den Netscape vom Vertragspartner Sun Microsystems lizenziert hat (siehe Lektion 28). Das war vermutlich ein Marketing-Trick, um im Fahrwasser der damals sehr erfolgreichen Sprache Java mitzuschwimmen. 2009 wurde Sun dann für 7,4 Milliarden US-Dollar von Oracle gekauft — dadurch ist nun Oracle Besitzer der Marke *JavaScript* Lektion 30.

Deswegen ist der Name *JavaScript* bei vielen Entwicklern nicht sonderlich beliebt. Erfahrene Entwickler sagen oft nur *JS*. Das gilt als »hip« und vermeidet den Java-Bezug (*Java* ist nicht gerade ein Synonym für »schlank und modern«).

JS auf dem Server

Statt im Browser kann JavaScript auch auf dem Server ausgeführt werden — zur dynamischen Generierung von Webseiten, als Alternative zu typischen Serversprachen wie PHP, Ruby oder Java. JavaScript auf dem Server auszuführen ist keine neue Idee. Eine ernstzunehmende Verbreitung hat die Sprache im Serverbereich aber erst seit ca. 2010/2011 erreicht, im Wesentlichen durch die Plattform Node.js[3].

Damit eröffnen sich für uns neue Möglichkeiten. Früher waren JS-Webentwickler sogenannte ***Frontender*** (Frontend-Enwickler), die sich vor allem mit der Optik und dem UI-Verhalten im Browser beschäftigt haben. Heutzutage können JS-Entwickler als ***FullStack***-Entwickler ihr Geld verdienen und beispielsweise komplexe Finanzregeln auf dem Server implementieren. Wo früher Webanwendungen zwei Programmiersprachen benötigten (JS auf dem Client und z.B. Java auf dem Server), sind heute komplette Anwendungen in JS geschrieben.

1. https://www.google.com/sheets/about
2. http://www.espruino.com/distance_sensing_robot
3. https://nodejs.org

JS im Embedded-Bereich

Schließlich liegen mit Espruino[4] und Tessel[5] zwei Boards vor, die JavaScript direkt auf der Hardware ausführen. Damit können Sie Ihre Pflanzen überwachen[6], Ihre Katze beobachten[7] oder einen Roboter programmieren[8]. Der Phantasie sind kaum Grenzen gesetzt.

Wir halten es nicht für übertrieben, zu behaupten:

JS ist heutzutage allgegenwärtig!

1.2 Was diese Class ist

Diese Class konzentriert sich auf JavaScript im Browser. Da es sich um Grundlagen handelt, können Sie das Erlernte (mit wenigen Ausnahmen) genauso auf dem Server verwenden — oder in PDFs, zum Scripten von Photoshop, für mobile Anwendungen, um Roboter zu steuern, um Kaffee zu kochen oder das Universum zu retten (beim letzten Punkt sind wir uns nicht zu 100% sicher).

Diese Class behandelt die Sprachversionen ***ES6***/***EcmaScript 2015*** und ***EcmaScript 2016*** von JavaScript. Mehr zu den JavaScript-Sprachversionen finden Sie in Lektion 28.

1.3 Was diese Class nicht(!) ist

Wenn Sie später Software mit JS entwickeln, werden Sie oft Details nachschlagen müssen. Dafür ist diese Class aber nur sehr bedingt geeignet. Wir versuchen Ihnen eher einen Leitfaden an die Hand zu geben, der Ihnen einen schnellen Einstieg in die Sprache JavaScript ermöglicht. Dabei ist die Reihenfolge der Lektionen auf den Lernfortschritt ausgerichtet, und nicht auf schnelles Auffinden von Themen optimiert. Wir erheben auch keinen Anspruch auf Vollständigkeit, sondern beschränken uns bewusst auf die Dinge, die für Sie als Einsteiger zunächst am wichtigsten sind — das »Big Picture« sozusagen.

Wenn Sie später bei Ihrer täglichen Entwicklungsarbeit das volle JS mit all seinen Details benötigen, empfehlen wir Ihnen folgende Nachschlagewerke:

- https://developer.mozilla.org — Mozilla Developer Network (MDN)
  Das MDN ist viel mehr als eine Referenz für den Firefox-Browser. Sie finden hier sowohl ausführliche Details zum JS-Sprachkern als auch zur Browser-Funktionalität. Selbst Tabellen, welcher Browser welches Feature ab welcher Version implementiert, sind vorhanden — häufig mit Polyfills (Code, der das Feature nachrüstet) für den Fall der Fälle.
- www.nodejs.org[9]
  Die offizielle Site zur Serverplattform Node.js. Ausführliche Dokumentation zum JS-Sprachkern und allen Node.js-spezifischen Erweiterungen.
- caniuse.com[10]
  Kann ich ein bestimmtes Feature benutzen? Diese Site kennt die Antwort. Ausführliche Übersichten zu neuen Features in HTML, CSS und JS mit Angaben zu den Browserversionen und Hinweisen zu Polyfills.

4. http://www.espruino.com
5. https://tessel.io/
6. https://www.hackster.io/ryanjgill2/plant-monitoring-system-88ed2b
7. https://www.hackster.io/girlie_mac/kittycam-powered-with-raspberry-pi-node-js-dd325b
8. https://www.hackster.io/29284/ble-bot-9000-c150b8
9. http://www.nodejs.org
10. http://caniuse.com

- **Speaking JavaScript** (Rauschmayer 2014, siehe Quellenverzeichnis)
  Dieses Buch gilt als *die* Referenz. Es ist modern, ausführlich und präzise.
- **JavaScript: The Definitive Guide** (Flanagen 2011, siehe Quellenverzeichnis)
  Das ist das JS-Standardwerk. Es ist zwar nicht mehr ganz aktuell, aber sehr gut strukturiert und ausführlich.
- **Understanding ECMAScript 6** (Zakas, siehe Quellenverzeichnis)
  Zakas Werk über alle Neuerungen von ES6/ES2015 ist bei Leanpub erhältlich, und außerdem als Open Source[11].

Falls Sie nach JS-Inhalten googeln möchten, ist es ratsam, ein »mdn« zu ergänzen, damit Sie gleich auf der passenden Seite des Mozilla Developer Networks landen — ansonsten besteht die Gefahr, dass Sie Seiten in der Trefferliste vorfinden, deren Erklärungen veraltet oder schlichtweg falsch sind.

1.4 Wo sind die Animationen und die GUI?

Wenn Sie schon einmal das Inhaltverzeichnis überflogen haben, ist Ihnen vielleicht aufgefallen, dass alle Themen recht technisch klingen und scheinbar kaum Bezug zur grafischen Oberfläche (GUI) haben. Die Screenshots sehen auch nicht aus, als könnte sich ein Künstler daran erfreuen. Ist das denn ernst gemeint?

Absolut — wir haben die Erfahrung gemacht, dass es einfacher ist, sich auf den Kern der Sprache zu konzentrieren, statt sich gleichzeitig mit hochglanzpolierten Animationen herumzuschlagen. Konzentrieren Sie sich aufs Wesentliche: auf die Sprache!

Wenn Sie die Sprache erst verstanden haben, ist der Rest einfach. Wie Sie HTML und CSS mit JS manipulieren, erfahren Sie deswegen erst im zweiten Band der Reihe. Wir hoffen, Ihnen den Einstieg dadurch etwas zu erleichtern. Der Code und die Konzepte, die Sie erlernen, sind deswegen nicht weniger praxisrelevant. Alle Beispiele und Übungen sind an Probleme und Situationen angelehnt, die wir tatsächlich so in unserem Programmieralltag erlebt haben.

1.5 Projekt »NerdWorld«

Willkommen zu Ihrem ersten Projekt. Natürlich ist es Ihr erklärtes Ziel, mit der Entwicklung von Software genug Geld zu verdienen, damit Sie sich baldmöglichst in die Karibik absetzen können. Dazu brauchen Sie aber Kunden — und diese sollten Sie optimal zufriedenstellen. Dann kommen weitere, die auch wieder Geld mitbringen.

Ein Kundengespräch

Stellen Sie sich vor, Ihr erster Kunde ist *Björn* — ein Ladenbesitzer, der seine Produkte nun auch im Internet anbieten möchte. Der Laden heißt »*NerdWorld*« (das Beispiel ist fiktiv — Sie können sich das Googeln sparen).

11. http://www.nczonline.net/blog/2014/03/26/announcing-understanding-ecmascript-6

NerdWorld ist ein Online-Shop, bei dem Nerds und andere Technik-Verrückte Produkte des »täglichen Bedarfs« kaufen: Nerfguns, Ufos mit USB-Anschluss, Klingonenwaffen fürs Büro, koffeinhaltige Getränke, stark koffeinhaltige Getränke, noch stärker koffeinhaltige Getränke usw. Eben alles, was der moderne Entwickler zum Überleben im Büro-Dschungel so braucht.

Es gibt zwar schon eine statische HTML-Site, nur können Sie dort bisher keine Produkte kaufen. Die neue Site soll primär aus einem Shop bestehen. Um das Ganze etwas interessanter zu gestalten, sind außerdem weitere Features geplant oder sollen weiterentwickelt werden, wie z.B.

- ein Chat-Client, über den sich die Kunden mit den Verkäufern/Kundenberatern austauschen
- eine Newsletter-Funktion, mit der der Shop die Kunden über neue Produkte informiert

Damit dieser Traum auch Realität wird, hat sich der Geschäftsführer an einen Spezialisten gewandt: an Sie! Sie treffen sich also mit ihm und er schildert Ihnen die Situation:

Seit Jahren verkaufen wir cooles Zeug für Nerds — oder solche, die es werden wollen. Unsere Produktpalette reicht von Star-Trek-Fanartikeln über Programmierer-Utensilien bis hin zu Smartphone-gesteuerten Mini-Drohnen und Scherzartikeln wie dem Elektroschock-Kugelschreiber.

Im Laufe dieses Kurses werden wir immer wieder auf Anforderungen von *Projekt NerdWorld* zu sprechen kommen und Ihnen zeigen, wie Ihnen JS dabei weiterhilft. Am Anfang werden das nur kleine Versatzstücke sein, die Sie leider noch nicht komplett in einem echten Projekt nutzen können. Zumindest sollte Ihnen der Bezug verdeutlichen, dass alles, was wir behandeln, relevant für die Praxis ist.

Welche der folgenden Aussagen sind richtig?

Bitte ankreuzen:

☐ JavaScript läuft nur im Browser.

☐ JavaScript kann auf Client und Server eingesetzt werden.

☐ Mit JavaScript lassen sich Roboter steuern.

☐ Die Bildbearbeitungssoftware Adobe Photoshop kann mit JavaScript gescripted werden.

☐ JavaScript wird häufig als Textauszeichnungssprache eingesetzt.

Los geht's: die »Chrome Webkonsole«

2

In dieser Lektion lernen Sie

- die Chrome Webkonsole kennen.
- Ihre ersten Zeilen Code zu schreiben.
- grundlegende Programmierrichtlinien kennen.

2.1 Wo ist denn die IDE?

Viele Entwickler schreiben ihren JS-Code in einer speziellen Entwicklungsumgebung, einer sogenannten ***IDE*** (Integrated Development Environment). Installation und Konfiguration ist allerdings mit viel Aufwand verbunden — den möchten wir Ihnen ersparen. Für Ihre ersten Schritte in JS benötigen Sie nicht viel. Es genügt, die aktuelle Version von ***Google Chrome*** zu installieren. Chrome bringt ausreichend Werkzeuge mit, um kleine JS-Anwendungen komplett zu entwickeln. Dazu zählt beispielsweise die ***Chrome Webkonsole***, die bereits viele Features mitbringt, die sonst eher IDEs vorbehalten sind.

Chrome erhalten Sie unter https://www.google.com/intl/de_de/chrome/ für Ihr jeweiliges Betriebssystem. Die Sprache können Sie auf der Seite umstellen — in diesem Kurs verwenden wir die englische Version (Vereinigte Staaten).

Schritt für Schritt 1:

1 Installieren und starten Sie Chrome auf Ihrem Betriebssystem.

2 Öffnen Sie die Chrome **Webkonsole** aus dem Developer-Menü (Abb. 3 4) oder per Tastaturkürzel: **ctrl-shift-j** bzw. **cmd-alt-j**.

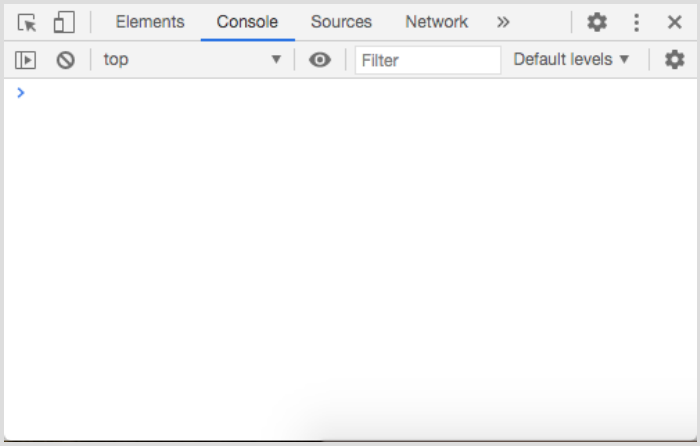


Abb. 3 *Google Chrome: Webkonsole*

3 Wenn Sie mehrere Zeilen schreiben möchten müssen Sie **shift-enter** drücken. Alleiniges drücken von **enter** führt den Code aus.

Alternativ können Sie sich auch ein Snippet erstellen. In einem Snippet können Sie Code in mehrere Zeilen schreiben. Wenn Sie Ihre Konsole bereits offen haben, dann können Sie über den Reiter *Source* in das Source-Menü wechseln. Innerhalb des Source-Menüs finden Sie auf der linken Seite weitere Reiter. Klicken Sie auf die zwei Pfeile nach rechts und wählen Sie *Snippets* aus.

Nun können Sie auf *New snippet* klicken und beliebig viele Snippets erstellen, in welche Sie ganz normal Code schreiben und diesen mit **ctrl-enter** bzw. **cmd-enter** ausführen können.

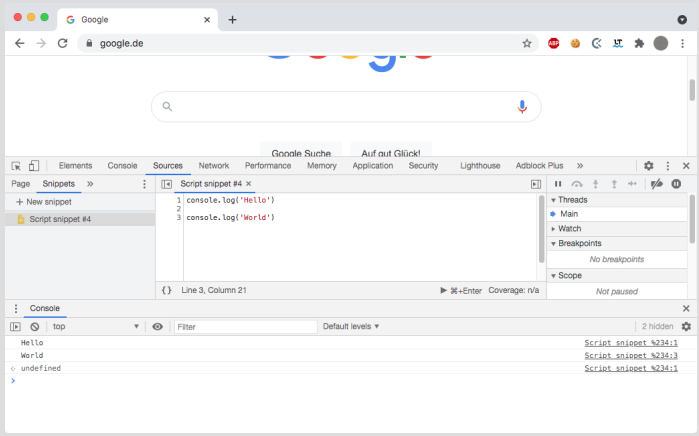


***Abb. 4** Google Chrome: Snippets*

2.2 Hilfreiche Shortcuts für die Webkonsole

Die folgende Tabelle zeigt einige hilfreiche Tastaturkürzel für die Webkonsole.

Windows/Linux	macOS	Funktion
ctrl-shift-j	**cmd-alt-j**	Konsole öffnen
enter	**enter**	Run — den Code in der Konsole ausführen

***Tabelle 2.1** Webkonsole Tastaturkürzel*

2.3 Hello World mit console.log

Traditionell beginnt das Erlernen einer neuen Programmiersprache mit der Ausgabe von `Hello World`. Sie müssen dazu lediglich die folgende Anweisung in der Webkonsole eingeben.

```
1 console.log('Hello World')
```

***Codebeispiel 1** accompanying_files/02/examples/hello.js*

Sobald Sie die Taste **enter** drücken, erscheint die Ausgabe in der nächsten Zeile.

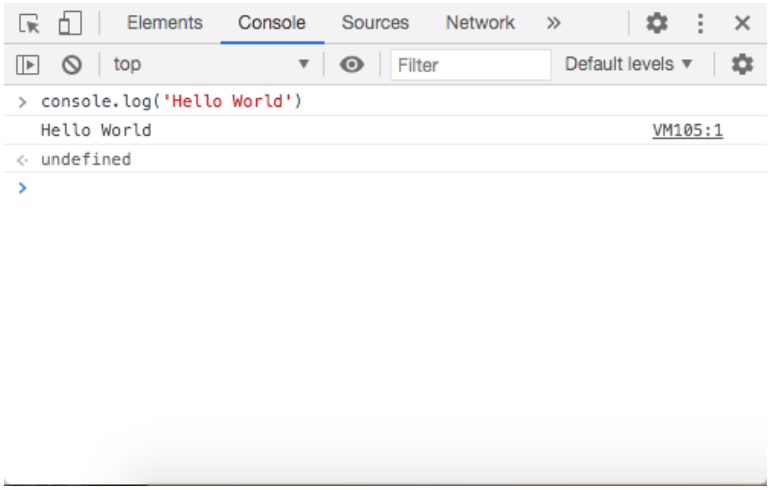


Abb. 5 *Ausgabe im Ausgabefenster*

Übung 1: In die Konsole geloggt

Geben Sie Ihren Namen aus.

2.4 Alarm! Alarm! — Die Konsole brennt

Betrachten Sie den JS-Code etwas genauer: `log` ist eine sogenannte ***Funktion***. Ihre Aufgabe ist es, in die Konsole (`console`) zu loggen, d.h. dort eine nachvollziehbare Ausgabe von Log-Meldungen einzutragen — wie in einem Logbuch.

Im Log lässt sich die Ausgabe des Programmes Schritt für Schritt nachvollziehen. Tatsächlich ist dieses Log nicht für Anwender bestimmt, sondern soll uns Programmierern helfen, den Ablauf eines Programmes nachzuvollziehen.

Ersetzen Sie `console.log` durch `alert`:

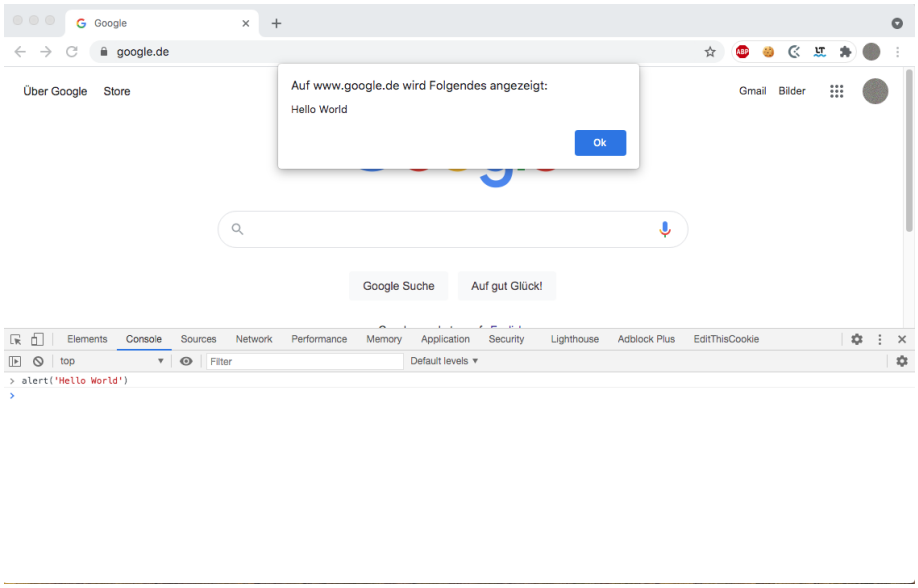


Abb. 6 *Ausgabe des Programms im Log*

Die Funktion `alert` öffnet eine sogenannte ***Alert-Box*** im Browserfenster. Sie ist dafür gedacht, Anwendern wichtige Warnmeldungen anzuzeigen. Die Alert-Box ist ein ***modales*** Popup-Fenster, d.h. der Anwender muss sie wegklicken, um die Website weiter benutzen zu können. Aus Usability-Gründen ist von solchen modalen Popup-Fenstern abzuraten. Setzen Sie sie nur in wenigen begründeten Fällen

ein. Außerdem handelt es sich dabei um eine Funktion, die nur in Browserumgebungen zur Verfügung steht. Wenn Sie z.B. auf dem Server unter *Node.js* programmieren, können Sie sie nicht verwenden.

Sowohl `alert` als auch `log` sind Funktionen. Den Aufruf einer Funktion erkennen Sie an den runden Klammern `()`. Innerhalb der Klammern hinterlegen Sie ein ***Argument*** — wie hier `'Hello World'`. Was mit dem Argument passiert, ist von der Funktion abhängig. Während `alert` einen Warnhinweis für den Anwender ausgibt, erzeugt `console.log` einen Logeintrag für uns Entwickler.

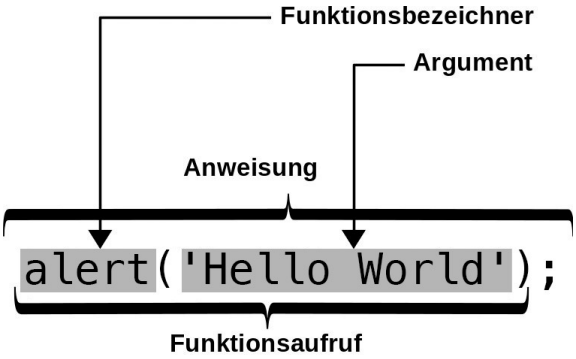


Abb. 7 *Begriffserläuterung zum Funktionsaufruf*

`alert` ist eine Funktion, die bereits im Browser implementiert ist. Der Browser bringt außerdem `console` mit, ein sogenanntes *Objekt*. Auf `console` steht Ihnen dann wiederum die Funktion `log` zu Verfügung. Vorerst werden Sie nur solche implementierten Funktionen verwenden. In Lektion 15 lernen Sie schließlich, Ihre eigenen Funktionen zu definieren.

Übung 2: Alarm!

Geben Sie den folgenden Text in einer Alert-Box aus: »*Alert! Your console is on fire! Hurry! Write an email to the firefighters or tweet with the hashtag #helpMyComputerIsOnFire*«

In dieser Class werden Sie der Einfachheit halber mit `console.log` arbeiten. In der Praxis gibt es weitere Möglichkeiten zur Ausgabe. Wir werden sie allerdings hier nicht behandeln, da sie nicht Teil des JS-Sprachkerns sind.

S-Words Shootout: Syntax oder Semantik?

Zwei wichtige Begriffe, denen Sie immer mal wieder in der Softwareentwicklung begegnen, sind ***Syntax*** und ***Semantik***. Es ist wichtig, den Unterschied zu verstehen. Beide Begriffe gibt es nicht erst bei Programmiersprachen. Sprachwissenschaftler prägten die Begriffe zunächst für natürliche Sprachen. Sie lassen sich aber durchaus mit der gleichen Bedeutung auf Programmiersprachen übertragen.

Bei der Syntax geht es um die Regeln, die den Aufbau von Sätzen einer Sprache beschreiben, oder im Fall von JavaScript den Aufbau von Anweisungen. Ist eine Anweisung syntaktisch falsch, meldet die JavaScript-Engine einen Syntaxfehler.

Beispiel

```
alert('Hello'
```

Codebeispiel 2

Hier fehlt die schließende runde Klammer. JavaScript quittiert das mit:

```
Uncaught SyntaxError: missing ) after argument list
```

Codebeispiel 3

Bei der Semantik dagegen geht es um die Bedeutung von Wörtern oder Sätzen. Im Fall von JavaScript also die Bedeutung des Codes. Was soll der Code bewirken?

Beispiel

Beispielsweise könnte der Auftraggeber sich eine Alert-Box mit einer Verabschiedung wünschen. Die Anweisung bewirkt aber eine Begrüßung:

```
1 alert('Welcome to our website')
```

Codebeispiel 4

Dann ist der Code zwar syntaktisch korrekt, aber semantisch falsch. Ein anderes Beispiel wäre ein Additionsprogramm, das falsch rechnet. Auch wenn es zwar grundsätzlich (im Sinne der JavaScript-Engine) funktioniert, so tut es dennoch das Falsche.

2.5 Mehrere Anweisungen

Selbstverständlich können Sie auch mehrere Anweisungen nacheinander loggen:

```
1 console.log('a log message')
2 console.log('... and another one')
```

Codebeispiel 5 *accompanying_files/02/examples/logmessage.js*

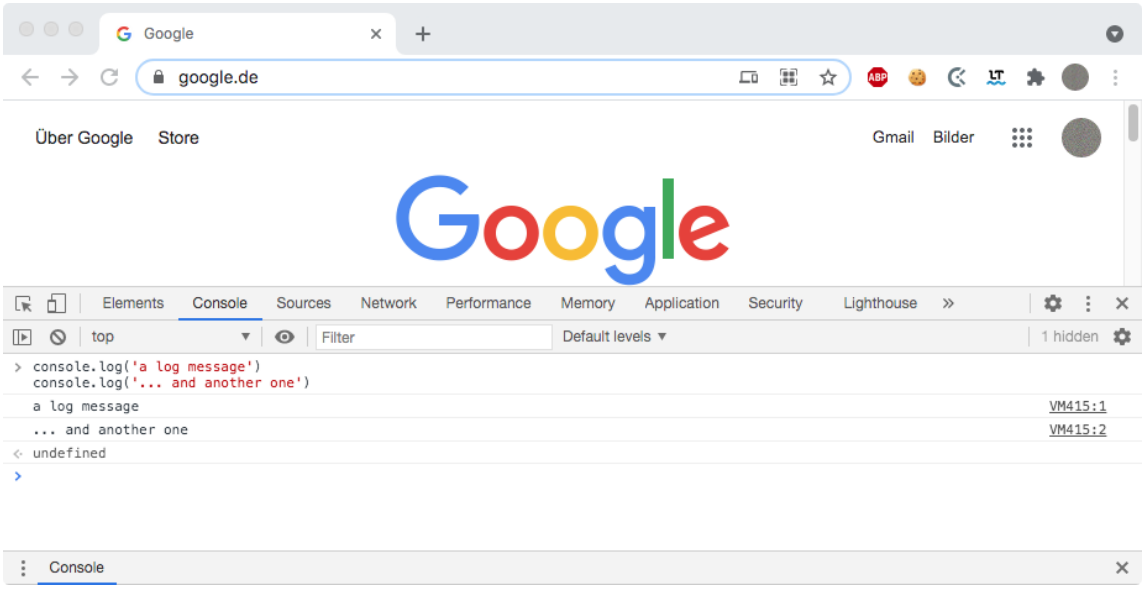


Abb. 8 *Zwei Ausgaben mit console.log*

Schreiben Sie jede Anweisung in eine eigene Zeile. Die Verwendung von Semikolon ist in JavaScript nicht nötig. Sie können das Semikolon also weglassen. In diesem Fall greift ein Mechanismus, der sich ***Automatic Semicolon Insertion*** (***ASI***) (dt.: automatische Semikolon-Ergänzung) nennt und das Semikolon für Sie automatisch setzt. Das ist einerseits recht praktisch, andererseits folgt die automatische Semikolon-Ergänzung relativ komplexen Regeln, die Sie erst einmal lernen sollten.

In der Praxis gibt es deswegen zwei Möglichkeiten, damit umzugehen:

1. Den semikolonlosen Stil
2. Den Semikolon-Stil

Beide Stile haben ihre Vor- und Nachteile und es gibt fanatische Verfechter auf beiden Seiten. Wir haben uns in dieser Class für den ersten Stil entschieden — aber nicht etwa, weil wir der Meinung wären, dass es sich dabei um den eindeutig besseren Stil handelt. Letztendlich ist das eher Geschmackssache. Was für den ersten Stil spricht, ist vor allem die größere Verbreitung.

Oder doch mal ohne?

Argumente und Hinweise der Schule des semikolonlosen Stils finden Sie bei:

- Marohnić 2010[12]
- Fuch 2013[13]
- Schlueter 2010[14]

Falls Sie den Umgang mit Semikolons üben möchten, können Sie Ihre Fähigkeiten gerne auf den ASI Scanning Test Grounds[15] unter Beweis stellen.

12. https://web.archive.org/web/20120501062915/http://mislav.uniqpath.com/2010/05/semicolons
13. https://github.com/madrobby/zepto/blob/master/CONTRIBUTING.md
14. http://blog.izs.me/post/2353458699/an-open-letter-to-javascript-leaders-regarding
15. http://asi.qfox.nl

2.6 Programmierrichtlinien

Wie Sie sich vielleicht schon vorstellen können, gibt es bei der Programmierung viele Freiheiten, z.B. wie Sie den Code formatieren, wo Sie Semikolons setzen usw. Damit nicht völliges Chaos ausbricht und Ihr Code konsistent bleibt, sollten Sie spezifische Regeln zur Schreibweise verwenden. Eine Sammlung solcher Regeln heißt ***Programmierrichtlinien***.

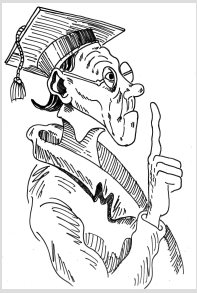
Softwarehäuser haben meist ihre eigenen Programmierrichtlinien, die sich von Firma zu Firma oder sogar von Projekt zu Projekt unterscheiden. Dabei ist vor allem entscheidend, dass Dinge einheitlich geregelt sind und jedes Team-Mitglied diese Regeln befolgt. Nur so lassen sich Wildwuchs und Chaos verhindern. Der exakte Wortlaut jeder Regel ist dagegen eher sekundär.

Gute Beispiele für ausführliche Programmierrichtlinien-Dokumente sind z.B. der jQuery-Style-Guide[16], den die Entwickler der jQuery-Bibliothek befolgen oder der JavaScript Styleguide von Airbnb[17].

Programmierrichtlinien sind auch wichtig, weil Sie nie wissen können, wo oder von wem Ihr Code einmal weiterentwickelt wird. Nachdem ein Projekt abgeschlossen ist, ist die entstandene Software oftmals noch jahrelang im Einsatz. Die Entwicklung steht nie still. Fehler sind zu verbessern, Änderungswünsche umzusetzen, Anpassungen durchzuführen — die sogenannte ***Wartung*** der Software. Der Programmierer, der den Code nach Ihnen pflegen, verbessern und abändern muss, ist Ihr ***Wartungsprogrammierer***. Seien Sie nett zu ihm! Helfen Sie ihm, Ihren Code zu verstehen, indem Sie sich an gängige Programmierrichtlinien halten. (Und wenn Ihnen nur nett sein nicht ausreicht, stellen Sie sich den Wartungsprogrammierer einfach als Serienkiller vor, der genau weiß, wo Sie wohnen…)
Vielleicht sind Sie selbst derjenige, der den Code Jahre später noch mal erweitern muss — seien Sie nett zu sich selbst!

> *Jeder Narr kann Code schreiben, den ein Computer versteht. Gute Programmierer schreiben Code, den Menschen verstehen.*
>
> Fowler (1999)

Wir werden ab sofort unsere eigenen Programmierrichtlinen entwickeln, dabei orientieren wir uns am JavaScript Styleguide von Airbnb[18]. Zu der jeweils aktuellen Thematik bestimmen wir passende Regeln, die uns das Programmiererleben erleichtern. Airbnb bietet seine Programmierrichtlinien[19] für JavaScript auf GitHub an. In dieser Sammlung finden Sie zu jedem Gebiet in JavaScript Anweisungen, wie Sie Ihren Code sauber schreiben. Folgende Regeln wollen wir ab sofort einhalten:



Programmierrichtlinien

- Schreiben Sie immer genau eine Anweisung in genau eine Zeile.
- Verzichten Sie beim Beenden einer Anweisung auf das Semikolon.

Weitere Programmierrichtlinien folgen in Kürze.

16. http://contribute.jquery.org/style-guide/js
17. https://github.com/airbnb/javascript
18. https://github.com/airbnb/javascript
19. https://github.com/airbnb/javascript

2.7 Seien Sie »strict«

Neben den Programmierrichtlinien gibt es noch eine weitere Möglichkeit, wie Sie sich selbst und Ihrem Wartungsprogrammierer das Leben etwas erleichtern können:

Seien Sie strikt!

Keine Sorge, dazu müssen Sie nicht gleich einen Besen verschlucken oder Ihre Kollegen mit strenger Stimme schimpfen, wenn Ihnen ihr Code nicht gefällt.

Es gibt vielmehr die Möglichkeit, JS in den sogenannten ***strict mode*** zu versetzen. In diesem Modus verhält sich JS tatsächlich strikter. Das bedeutet:

- Einige Richtlinien, die der *standard mode* als reine Konvention betrachtet, führen im *strict mode* zu einer Fehlermeldung.
- Viele veraltete Sprachkonstrukte werden nicht mehr akzeptiert (z.B. die `with`-Anweisung)
- Fehlerhafter oder problematischer Code, den JS im standard mode ohne Murren akzeptiert, führt nun zu einer (meist) aussagekräftigen Fehlermeldung. Damit bemerken Sie Probleme frühzeitig und sparen sich die oft aufwendige Fehlersuche.
- Einige der neueren Sprachkonstrukte (z.B. `let`) lassen sich in den meisten Umgebungen überhaupt nur im strict mode verwenden.

Um JS in den strict mode zu versetzen, müssen Sie nicht viel tun: Es genügt, wenn Sie die Anweisung

```
1 'use strict'
```

Codebeispiel 6

an den Anfang Ihrer JS-Datei schreiben.

Ab sofort werden wir das immer tun, da es für uns praktisch keine Nachteile hat. Wir müssen schließlich keinen Legacy-Code (genauer: Code in alter JS-Syntax) unterstützen. In Lektion 20 erfahren Sie dann noch mal genauer, was es mit `'use strict'` auf sich hat. Wenn Sie es noch genauer wissen möchten, finden Sie auf MDN (Mozilla Development Network) einen Eintrag über Strict Mode[20], der alle Unterschiede zwischen den Modi erläutert.

2.8 Kommentare oder: Geben Sie doch Ihren Senf dazu

Mit *Kommentaren* ist es möglich, hilfreiche Erläuterungen zum Code zu schreiben oder auch Codezeilen zu deaktivieren. Kommentare können helfen, Code schneller zu verstehen, den Ihr früheres Ich oder ein anderer Programmierer geschrieben hat.

Richtig platzierte Kommentare verbessern die Lesbarkeit Ihres Codes.

Der JavaScript-Interpreter ignoriert alles, was sich in Kommentaren befindet. Das gilt sogar für vollständige JavaScript-Anweisungen. In JavaScript gibt es zwei Möglichkeiten, Kommentare einzubinden. Dazu wird zwischen

- *einzeiligen* und
- *mehrzeiligen* Kommentaren

unterschieden.

20. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict_mode/Transitioning_to_strict_mode#Differences_from_non-strict_to_strict

Einzeilige Kommentare binden Sie mit zwei Slashes `//` ein.

```
1 //The following code contains a meaningful string.
2 console.log('I am a meaningful string')
```

Codebeispiel 7

Sie sollten die Kommentare in der Praxis nur für komplexere Zusammenhänge verwenden und keine Trivialitäten kommentieren, wie im obigen Beispiel. Wenn Sie aufgrund zu vieler Kommentare Ihren Code kaum noch erkennen können, haben Sie nichts gewonnen.

Mehrzeilige Kommentare erlauben Ihnen, längere — eben mehrzeilige — Texte in den Code einzubetten. Ein solcher Kommentar beginnt mit den Zeichen `/*` und endet mit `*/`.

```
1 /*
2   TODO: Add a more spicy greeting here. Candidates are:
3    - Speak Friend and Enter
4    - Live long and prosper
5    - Welcome to the Future (in mystery Futurama Voice)
6 */
7 alert('Welcome to NerdWorld')
```

Codebeispiel 8 *accompanying_files/02/examples/comment.js*

Kommentare, die mit `//` beginnen, müssen am Ende einer Zeile stehen oder komplett alleine in der Zeile. Kommentare der Form `/* ein Kommentar */` dürfen Sie dagegen fast überall platzieren.

Beispiel

```
1 console.log(/* some comment inside */ 'I am a meaningful string')
```

Codebeispiel 9

2.9 Testen Sie Ihr Wissen!

1. Welche der folgenden Zeilen sind syntaktisch korrekt?

   Bitte ankreuzen:

   - ☐ console.log 'Hello'
   - ☐ console.log("Hello")
   - ☐ console.log 'Hello';
   - ☐ console.log('Hello')

2. Wie versetzen Sie JS in den strikten Modus?

   Bitte ankreuzen:

   - ☐ Durch `'use strict'` am Anfang des Codes.
   - ☐ Durch den Aufruf der Funktion `useStrict()` am Anfang des Codes.
   - ☐ Starten Sie Chrome mit der Kommandozeilen-Option `--strict`.
   - ☐ Durch `'strict'` am Anfang des Codes.
   - ☐ Wählen Sie *strict mode* aus dem *Developer*-Menü des Browsers.

Inhalt von b: 1

Inhalt von c: 2

2. **Bestimmen Sie den Inhalt von `a` und `b`, nach dem Ausführen der folgenden Anweisung:**

```
1 const [b, a] = [3, 2, 1]
```

Inhalt von a: 2

Inhalt von b: 3

3. **Bestimmen Sie den Inhalt von `a` und `b`, nach dem Ausführen der folgenden Anweisung:**

```
1 const [b, a] = [1, 2, 3]
```

Inhalt von a: 2

Inhalt von b: 1

Index

R

S

T

U

V

W

Z