

HANS-GEORG SCHUMANN

JAVASCRIPT

FÜR **KIDS**

PROGRAMMIEREN LERNEN
OHNE VORKENNTNISSE



INHALT

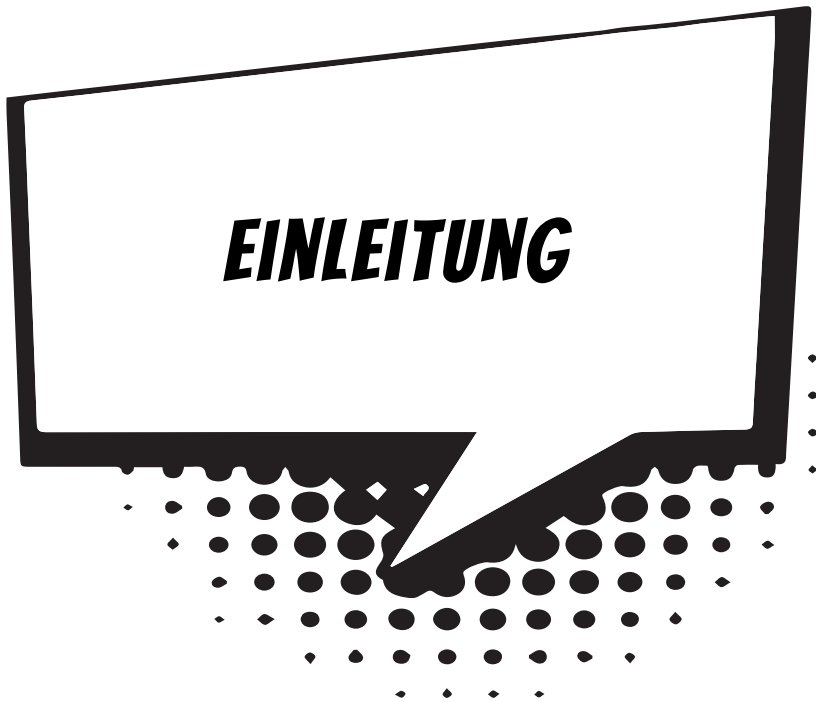
EINLEITUNG	11	
Was heisst Programmieren?	12	
Was ist eine Entwicklungsumgebung?	12	
Warum gerade JavaScript?	13	
Die Entwicklungsumgebung	13	
Wie arbeite ich mit diesem Buch?	14	
Was brauchst du für dieses Buch?	15	
Hinweise für Lehrer	16	
ALLER ANFANG IST SCHWER? EINSTIEG IN JAVASCRIPT	17	1
Ein erstes Hallo, schlicht und einfach	17	
Visual Studio Code starten	21	
Die nötigen Erweiterungen	25	
Der steinige Weg zum zweiten Hallo	28	
Visual Studio Code beenden	32	
Zusammenfassung	33	
Ein paar Fragen	34	
... aber noch keine Aufgabe	34	
WENN UND DANN UND MATHE: BEDINGUNG UND KONTROLLE	35	2
Willkommen in JavaScript	36	
Die if-Struktur	41	
if und else	46	
Ein bisschen Grundrechnen	49	
Was für Zahlen?	52	
Zusammenfassung	55	
Ein paar Fragen	56	
... und ein paar Aufgaben	56	

3	ZENSUREN UND ZAHLENRATEN: VERGLEICHEN UND WIEDERHOLEN	57
	Zensurenbild	58
	Punkt für Punkt	62
	Ein kleines Ratespiel	66
	Dein Computer zählt mit	71
	Noch mehr Spielkomfort	73
	Zusammenfassung	74
	Ein paar Fragen	75
	... und ein paar Aufgaben	75
4	GELD-SPIELEREIEN: NOCH MEHR SCHLEIFEN	77
	Spiel mit dem Glück	77
	Die for-Struktur	80
	Auf dem Weg zum Millionär	82
	Macht Lotto wirklich reich?	89
	Zeichen-Verkettung	93
	Zusammenfassung	95
	Ein paar Fragen	96
	... und ein paar Aufgaben	96
5	MARKE EIGENBAU: FUNKTIONEN	97
	JavaScript ist lernfähig	97
	Lokal oder global?	101
	Tauschprozesse	104
	Zahlen sortieren	109
	Zusammenfassung	112
	Ein paar Fragen	112
	... und ein paar Aufgaben	112
6	FRANKENSTEINS LABOR: KLASSEN UND MODULE	113
	Ein neues Baby?	114
	Vererbung	118
	Programm-Module	123
	Zusammenfassung	128
	Ein paar Fragen	129
	... aber keine Aufgabe	129

MEHR ÜBER HTML: BUTTONS UND LABELS	131	7
Ein Gerüst in HTML	132	
Es passiert etwas	136	
Noch mal Hallo	140	
Noch etwas Feinschliff	144	
Zusammenfassung	147	
Ein paar Fragen	148	
... und eine Aufgabe	148	
HALLO, WIE GEHT'S? KOMPONENTENSAMMLUNG	149	8
Kleine Button-Parade	149	
Listenwahl	152	
Von Pünktchen	156	
... und Häkchen	160	
Zusammenfassung	162	
Ein paar Fragen	163	
... und ein paar Aufgaben	163	
WER WEISS WAS? QUIZ-PROJEKT TEIL 1	165	9
Zuerst der Plan und dann der Bau	165	
Spielbereit?	170	
Datentransfer	172	
Auf ein Neues	176	
Zusammenfassung	180	
Ein paar Fragen	180	
... aber keine Aufgabe	180	
SPIELEN UND LERNEN: QUIZ-PROJEKT TEIL 2	181	10
Richtig oder falsch?	181	
Aufgabenkontrolle	186	
Antwort als Optionen	190	
Vokabeln lernen?	192	
Zusammenfassung	196	
Ein paar Fragen	197	
... und ein paar Aufgaben	197	

11	JETZT WIRD ES BUNT: GRAFIK IN JAVASCRIPT	199
	Von Punkten und Koordinaten	200
	Das erste Bild	202
	Jetzt wird's bunt	205
	Eckig und rund	208
	Mit Text geht auch	209
	Farbtupfer	211
	Selber zeichnen?	212
	Zusammenfassung	216
	Ein paar Fragen	217
	... und ein paar Aufgaben	217
12	BILDER LERNEN LAUFEN: ANIMATIONEN	219
	Erst mal (irgend)ein Bild	219
	Kommen und gehen	221
	Bildersammlung	226
	Drehungen	232
	Zusammenfassung	234
	Ein paar Fragen	235
	... und ein paar Aufgaben	235
13	BUNTES TRIO: KLEINE SPIELESAMMLUNG	237
	Wie viele Augen?	237
	Schere – Stein – Papier	240
	Auf ein Wort	244
	Angst vor dem Galgen?	247
	Komplettierung	251
	Zusammenfassung	254
	Keine Fragen	255
	... aber ein paar Aufgaben	255
14	VON DER KUGEL ZUM PLAYER: EIN KLEINES BALLSPIEL TEIL 1	257
	Ein Ball auf dem Spielfeld	258
	Grenzkontrollen	261
	Player-Klasse	264
	Zwei Paddles	267
	Der Computer spielt mit	271
	Zusammenfassung	273
	Ein paar Fragen	273
	... und eine Aufgabe	273

PING UND PONG: EIN KLEINES BALLSPIEL TEIL 2	275	15
Spiel-Intelligenz?	275	
Manuelle Paddle-Steuerung	280	
Feintuning	283	
Das Gesamt-Listing	286	
Zusammenfassung	290	
Keine Fragen	290	
... und keine Aufgaben	290	
SCHLANGE UND KÄFER: DAS SNAKE-SPIEL	291	16
Ein Feld, zwei Spieler	291	
Die Schlange bewegt sich	296	
Länger und länger	299	
Verfeinerungen	303	
Das gesamte Listing	305	
Zusammenfassung	308	
Eine Frage	308	
... und eine Aufgabe	308	
TRAU DICH WAS: DODGE ODER HIT?	309	17
Alter Quelltext, neue Idee?	309	
Stand, Duck, Jump	313	
Ausweichmanöver	316	
Zusammenfassung	322	
Keine Fragen	322	
... und nur eine Aufgabe	322	
ANHANG A VISUAL STUDIO CODE INSTALLIEREN	323	A
ANHANG B KLEINE CHECKLISTE	329	B
STICHWORTVERZEICHNIS	331	



Computer sind schon wahre Wunder-Maschinen. Doch das sind sie nur dadurch, dass es schlaue Programmierer gibt, die die passenden Anwendungen und Spiele erstellen. Wenn du eines Tages zur Gilde der Programmierer zählen willst, dann hast du hier die Möglichkeit, bei null anzufangen – ohne irgendwelche Programmierkenntnisse.

Hier lernst du, in JavaScript zu programmieren, von dem du sicher schon einmal etwas gehört hast. Das ist nicht die kleine Schwester der Programmiersprache Java. Was ist der Unterschied? Grob gesagt: Es gibt eine Gruppe von Sprachen, da wird aus dem, was du programmierst, direkt ein ausführbares Programm erzeugt. Dazu gehören Java, C#, C++. Und es gibt Sprachen, da ist immer ein sogenannter Interpreter nötig, der das Programm ausführt. Eine solche Sprache ist JavaScript, und der passende Interpreter steckt in fast jedem Webbrowser.

Du hast bestimmt auch schon mal von HTML gehört. Ohne diese »HyperText Markup Language« könntest du dir keine Webseiten anschauen. HTML dient dazu, elektronische Dokumente zu erstellen. Am bekanntesten wurde HTML als das Format für Web-Dokumente, wie z.B. Seiten von Homepages, die sich in einem passenden Browser darstellen lassen.

Und JavaScript, das Mitte der 1990er Jahre entwickelt wurde, erweitert die Möglichkeiten von HTML stark. Mithilfe von JavaScript lassen sich sogar Spiele erstellen, die man direkt im Browser spielen kann.

JavaScript hat viel von anderen (großen) Programmiersprachen übernommen. Diese Sprache ist leicht erlernbar, und weil JavaScript-Programme direkt in fast jedem Browser funktionieren, braucht man nicht viel Werkzeug, um Anwendungen und Spiele zu erstellen.

WAS HEISST PROGRAMMIEREN?

Wenn du aufschreibst, was ein Computer tun soll, nennt man das **Programmieren**. Das Tolle daran ist, dass du selbst bestimmen kannst, was getan werden soll. Lässt du dein Programm laufen, macht der Computer das, was du ausgeheckt hast. Natürlich wird er nicht dein Zimmer aufräumen und dir auch keine Tasse Kakao ans Bett bringen. Aber beherrschst du erst mal das Programmieren, kannst du den Computer sozusagen nach deiner Pfeife tanzen lassen.

Allerdings passiert es gerade beim Programmieren, dass der Computer nicht so will, wie du es gerne hättest. Meistens ist das ein Fehler im Programm. Das Problem kann aber auch irgendwo anders im Computer oder im Betriebssystem liegen. Das Dumme bei Fehlern ist, dass sie sich gern so gut verstecken, dass die Suche danach schon manchen Programmierer zur Verzweiflung gebracht hat.

Vielleicht hast du nun trotzdem Lust bekommen, das Programmieren zu erlernen. Dann brauchst du ja nur noch eine passende Entwicklungsumgebung, und schon kann es losgehen.

WAS IST EINE ENTWICKLUNGSUMGEBUNG?

Um ein Programm zu erstellen, musst du erst einmal etwas eintippen. Das ist wie bei einem Brief oder einer Geschichte, die man schreibt. Das Textprogramm dafür kann sehr einfach sein, weil es ja nicht auf eine besondere Schrift oder Darstellung ankommt wie bei einem Brief oder einem Referat. So etwas wird **Editor** genannt.

Ist das Programm eingetippt, kann es der Computer nicht einfach lesen und ausführen. Jetzt muss es so übersetzt werden, dass der PC versteht, was du von ihm willst. Weil er aber eine ganz andere Sprache spricht als du, muss ein Dolmetscher her.

Ein solcher Dolmetscher (= Interpret) steckt in jedem Browser, JavaScript funktioniert sogar unter mehreren Betriebssystemen. Dein Computer kann also ein Windows-PC oder ein Linux-PC sein, ein Android-Smartphone oder irgendein anderes System. Ein und dasselbe JavaScript-Programm kann so (eventuell mit kleinen Abweichungen) auf fast jedem Computer oder Smartphone funktionieren.

Schließlich müssen Programme getestet, überarbeitet, verbessert, wieder getestet und weiterentwickelt werden. Da ist man froh, wenn man einige zusätzliche Hilfen hat. Daraus wird dann ein ganzes System, die Entwicklungsumgebung.

WARUM GERADE JAVASCRIPT?

Leider kannst du nicht so programmieren, wie dir der Schnabel gewachsen ist. Eine **Programmiersprache** muss so aufgebaut sein, dass möglichst viele Menschen in möglichst vielen Ländern einheitlich damit umgehen können.

Weil in der ganzen Welt Leute zu finden sind, die wenigstens ein paar Brocken Englisch können, besteht auch fast jede Programmiersprache aus englischen Wörtern. Es gab auch immer mal Versuche, z.B. in Deutsch zu programmieren, aber meistens klingen die Wörter dort so künstlich, dass man lieber wieder aufs Englische zurückgreift.

Eigentlich ist es egal, welche Programmiersprache du benutzt. Am besten eine, die möglichst leicht zu erlernen ist. Wie du weißt, bekommst du es in diesem Buch mit der Programmiersprache JavaScript zu tun, die mittlerweile sehr weit verbreitet ist. (Willst du mal in andere Sprachen hineinschnuppern, dann empfehle ich dir z.B. eines der anderen Kids-Bücher über Python, Java oder C++.)

Der Weg zum guten Programmierer kann ganz schön steinig sein. Nicht selten kommt es vor, dass man die Lust verliert, weil einfach gar nichts klappen will. Das Programm tut etwas ganz anderes, man kann den Fehler nicht finden und man fragt sich: Wozu soll ich eigentlich programmieren lernen, wo es doch schon genug Programme gibt?

Gute Programmierer werden immer gesucht, und dieser Bedarf wird weiter steigen. Und JavaScript gehört dabei durchaus zu den erwünschten Sprachen. Wirklich gute Programmierer werden auch wirklich gut bezahlt. Es ist also nicht nur einen Versuch wert, es kann sich durchaus lohnen, das Programmieren in JavaScript zu erlernen.

DIE ENTWICKLUNGSUMGEBUNG

Um eine Entwicklungsumgebung für JavaScript musst du dich nicht weiter kümmern, wenn dir eine einfache reicht. Das ist der Editor, der mit einem Betriebssystem wie Windows mitinstalliert wird. Besser aber ist etwas mit mehr Komfort, wie Visual Studio Code von Microsoft. Dieses Paket werden wir hier ausgiebig benutzen. Du kannst es dir von dieser Seite herunterladen:

<https://code.visualstudio.com/>

(Im Anhang steht, wie du dieses Paket installierst, und im Buch wird erklärt, wie du es nutzen kannst.)

UND WAS BIETET DIESES BUCH?

Über eine ganze Reihe von Kapiteln verteilt lernst du

- ⊙ das Basiswissen von JavaScript kennen
- ⊙ einiges über HTML und DOM
- ⊙ etwas über objektorientierte Programmierung
- ⊙ die grafischen Möglichkeiten von JavaScript kennen
- ⊙ wie man eigene Game- und Player-Module programmiert

Im Anhang gibt es dann noch zusätzliche Informationen und Hilfen, unter anderem über Installationen und den Umgang mit Fehlern.

WIE ARBEITE ICH MIT DIESEM BUCH?

Grundsätzlich besteht dieses Buch aus einer Menge Text mit vielen Abbildungen dazwischen. Natürlich habe ich mich bemüht, alles so zuzubereiten, dass daraus lauter gut verdauliche Happen werden. Damit das Ganze noch genießbarer wird, gibt es zusätzlich noch einige Symbole, die ich dir hier gern erklären möchte:

ARBEITSSCHRITTE

➤ Wenn du dieses Zeichen siehst, heißt das: Es gibt etwas zu tun. Damit kommen wir beim Programmieren Schritt für Schritt einem neuen Ziel immer näher.

Grundsätzlich lernt man besser, wenn man einen Programmtext selbst eintippt oder ändert. Aber nicht immer hat man große Lust dazu. Deshalb gibt es alle Projekte im Buch auch als Download:

<http://www.mitp.de/0263>

Und hinter einem Programmierschritt findest du auch den jeweiligen Namen des Projekts oder eines Ordners (z.B. → PROJEKT1). Wenn du also das Projekt nicht selbst erstellen willst, kannst du stattdessen die passenden Dateien laden (sie liegen alle im Haupt-Ordner PROJEKTE).

AUFGABEN

Am Ende eines Kapitels findest du jeweils eine Reihe von Fragen und Aufgaben. Diese Übungen sind nicht immer ganz einfach, aber sie helfen dir, noch besser zu

WAS BRAUCHST DU FÜR DIESES BUCH?

programmieren. Lösungen zu den Aufgaben findest du in verschiedenen Formaten ebenfalls im Verzeichnis PROJEKTE. Du kannst sie dir alle im Browser oder in einem Textverarbeitungsprogramm anschauen. Oder du lässt sie dir ausdrucken und hast sie dann schwarz auf weiß, um sie neben deinen Computer zu legen. (Auch die Programme zu den Aufgaben liegen im Ordner PROJEKTE.)

NOTFÄLLE

Vielleicht hast du irgendetwas falsch gemacht oder etwas vergessen. Oder es wird gerade knifflig. Dann fragst du dich, was du nun tun sollst. Bei diesem Symbol findest du eine Lösungsmöglichkeit. Notfalls kannst du aber auch ganz hinten im Anhang B nachschauen, wo einige Hinweise zur Pannenhilfe aufgeführt sind.



WICHTIGE STELLEN IM BUCH

Hin und wieder findest du ein solch dickes Ausrufezeichen im Buch. Dann ist das eine Stelle, an der etwas besonders Wichtiges steht.



Wenn du ein solches »Wow« siehst, geht es um ausführlichere Informationen zu einem Thema.



WAS BRAUCHST DU FÜR DIESES BUCH?

Installiert wird Visual Studio Code mit einem Setup-Programm in ein Verzeichnis deiner Wahl. Außerdem solltest du einen Ordner einrichten, in dem du später deine JavaScript-Projekte unterbringen kannst – z.B. D:\JSCRIPT.

Die Beispielprogramme in diesem Buch gibt es alle als Download von der Homepage des Verlages, falls du mal keine Lust zum Abtippen hast:

<http://www.mitp.de/0263>

Und auch die Lösungen zu den Fragen und Aufgaben sind dort untergebracht (alles im Ordner PROJEKTE).

BETRIEBSSYSTEM

Die meisten Computer arbeiten heute mit dem Betriebssystem Windows. Davon brauchst du eine möglichst neue Version (am besten von Windows 10). JavaScript gibt es unter anderem auch für Linux, aber hier im Buch geht es nur um Windows.

SPEICHERMEDIEN

Auf jeden Fall benötigst du etwas wie einen USB-Stick oder eine SD-Card, auch wenn du deine Programme auf die Festplatte speichern willst. Auf einem externen Speicher sind deine Arbeiten auf jeden Fall zusätzlich sicher aufgehoben. Bitte gebenebenfalls deine Eltern oder Lehrer um Hilfe.

HINWEISE FÜR LEHRER

Dieses Buch versteht sich auch als Lernwerk für den Informatik-Unterricht in der Schule. Dort setzt natürlich jeder Lehrer seine eigenen Schwerpunkte. Benutzen Sie an Ihrer Schule bereits ein Werk aus einem Schulbuchverlag, so lässt sich dieses Buch auch als Materialienband einsetzen – in Ergänzung zu dem vorhandenen Schulbuch. Weil dieses Buch sozusagen »bei null« anfängt, ist ein direkter Einstieg in JavaScript möglich – ohne irgendwelche anderen Programmierkenntnisse.

Ein wichtiger Schwerpunkt in diesem Buch ist die objektorientierte Programmierung (OOP). Auf die wichtigsten Eigenheiten (Kapselung, Vererbung und Polymorphie) wird ausführlich eingegangen. Ein großer Schwerpunkt ist die Programmierung von Spielen.

In den Projekten werden alle wesentlichen Elemente des Wortschatzes von HTML und JavaScript wie auch die wichtigsten Grafik-Komponenten eingesetzt. In den Lösungen zu den Aufgaben finden Sie weitere Vorschläge zur Programmierung.

ÜBUNGSMEDIEN

Für den Informatik-Unterricht sollte jeder Schüler ein eigenes externes Speichermedium haben, um darauf seine Programmierversuche zu sichern. So wird verhindert, dass sich auf der Festplatte des Schulcomputers mit der Zeit allerlei »Datenmüll« ansammelt. Außerdem dient der eigene Datenträger dem Datenschutz: Nur der betreffende Schüler kann seine Daten manipulieren.

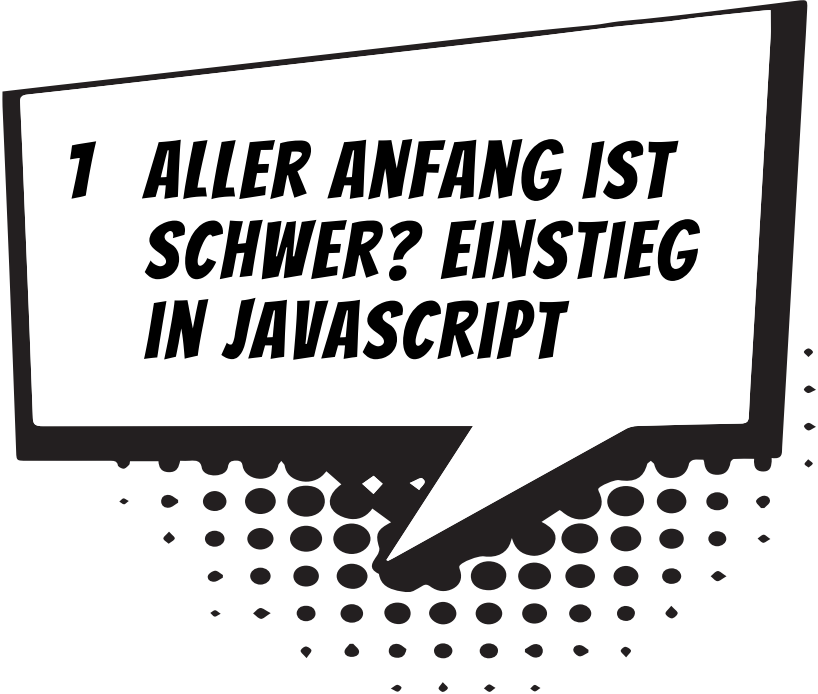
AUF DIE DATEIEN ZUM BUCH VERZICHTEN?

Vielleicht ist es Ihnen lieber, wenn Ihre Schüler die Projekte alle selbst erstellen. Dann lassen Sie die Download-Dateien einfach (erst einmal) weg.

REGELMÄßIG SICHERN

Es kann nicht schaden, die Programmdateien, an denen gerade gearbeitet wird, etwa alle zehn Minuten zu speichern. Denn Computer pflegen gern gerade dann »abzustürzen«, wenn man seine Arbeit längere Zeit nicht gespeichert hat.

1 ALLER ANFANG IST SCHWER? EINSTIEG IN JAVASCRIPT



Du willst gleich loslegen? Dem Computer endlich mal etwas sagen, was er für dich tun kann? Na, dann schalte deinen PC an und lass erst mal Windows auftauchen. Von da aus geht es dann direkt zum ersten Programmprojekt in JavaScript.

In diesem Kapitel lernst du

- ⊙ etwas über HTML
- ⊙ wie man ein kleines Programm mit dem Editor schreibt
- ⊙ eine Anweisung für die Ausgabe kennen
- ⊙ wie der Browser das Programm ausführen kann
- ⊙ wie man ein neues Projekt in Visual Studio Code erstellt
- ⊙ etwas über die Datei LAUNCH.JSON
- ⊙ wie man Visual Studio Code beendet

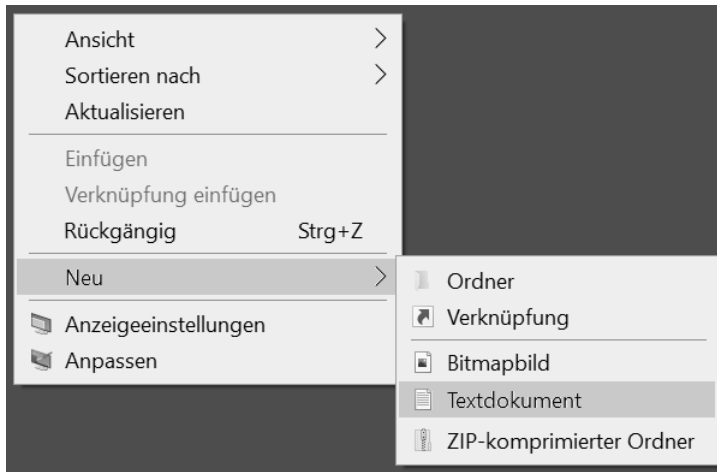
EIN ERSTES HALLO, SCHLICHT UND EINFACH

Eigentlich ist es ganz einfach, ein erstes Programm in JavaScript zu schreiben. Du brauchst dazu nur einen simplen Editor wie den, der stets mit Windows mitinstal-

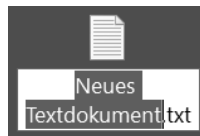
liert wird, auch Notepad genannt. Und du brauchst einen Browser, das ist das Programm, mit dem du im Internet surfst oder Dateien von dort herunterlädst. Also z.B. Chrome von Google oder Edge von Microsoft, um nur zwei zu nennen.

Probieren wir aus, wie wir den Browser zum Hallo-Sagen bringen.

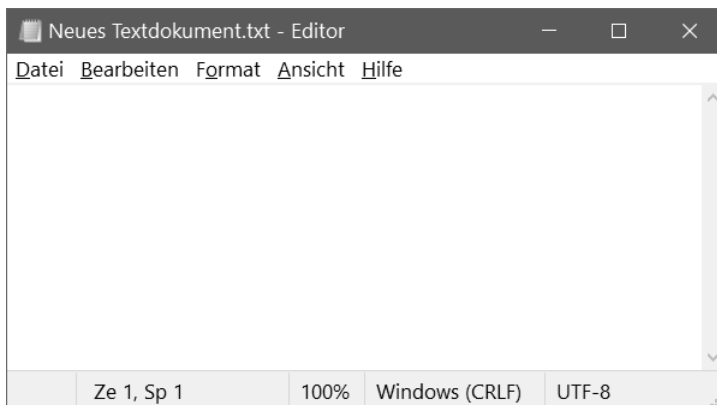
- Klicke mit der rechten Maustaste auf den Desktop, ein Kontextmenü öffnet sich, klicke dort auf NEU und dann auf TEXTDOKUMENT.



Auf dem Desktop findest du nun ein neues Symbol.

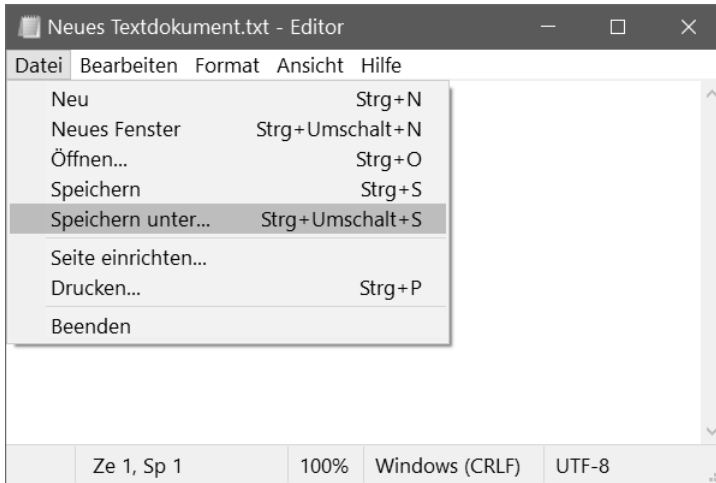


- Wenn du auf dieses Symbol doppelklickst, öffnet sich kurze Zeit später der zugehörige Editor und bietet dir ein leeres Fenster an.

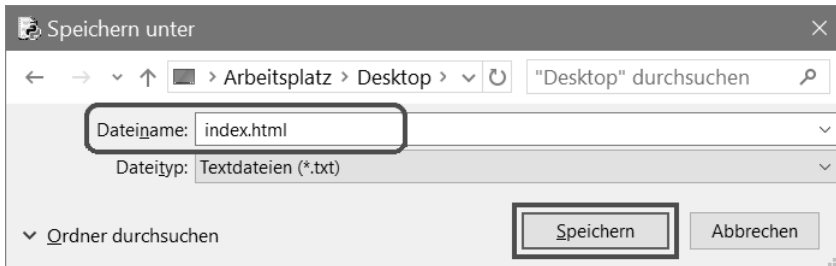


Bevor wir da etwas hineintippen, speichern wir die noch leere Datei unter einem passenden Namen:

➤ Klicke im Editor oben auf DATEI und dann auf SPEICHERN UNTER.



➤ Gib der Datei den Namen INDEX.HTML und klicke dann auf SPEICHERN.



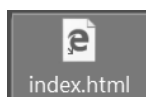
Warum wir die Datei so genannt haben, erkläre ich später. Jetzt wollen wir sie mit Inhalt füllen. Aber nur mit einigen wenigen Zeilen.

➤ Tippe also ein:

```
<script>
document.write("Hallo");
</script>
```

➤ Dann speichere das Ganze – z.B. über DATEI und SPEICHERN oder mit `[Strg] + [S]`.

Und nun kommen wir zu dem, was du als Symbol auf dem Desktop sehen kannst:





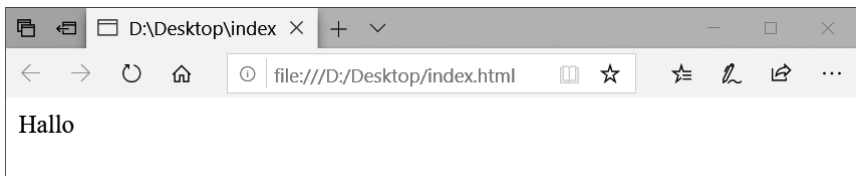
Warum INDEX und warum HTML? Die Kennung HTML ist die Abkürzung für »HyperText Markup Language«, eine Hilfssprache für das Programmieren von Webseiten.

Was konkret heißt, dass unser kleines Programm da oben zum Teil in HTML geschrieben wurde. Denn eigentlich ist HTML eine Beschreibungssprache, die den Aufbau einer Webseite beschreibt.

INDEX heißt unsere Datei, weil sie so vom Browser als Startdatei einer vermeintlichen Homepage bzw. Webseite interpretiert wird. Das siehst du gleich nach dem Start.

Wie das Dateisymbol für INDEX.HTML aussieht, hängt davon ab, welcher Browser dein Standard-Browser ist. (Ich benutze außer Microsoft Edge auch Google Chrome und Mozilla Firefox.)

➤ Doppelklicke auf das Symbol mit dem Namen INDEX.HTML.



Der Browser (mit dem du normalerweise im Internet surfst) öffnet sich und zeigt ein einfaches, aber nettes »Hallo«. Womit du dein erstes Programm geschafft hast – allerdings nicht in reinem JavaScript. Das ist ja die Sprache, um die es in diesem Buch eigentlich gehen soll. Lediglich die Zeile in der Mitte ist in JavaScript geschrieben, das Drumherum ist aber nötig, damit der Browser weiß, dass es um JavaScript geht.

Jeder Internet-Browser versteht die Sprache JavaScript, weil er einen eingebauten Interpreter für diese Sprache hat. Ein Interpreter ist eine Art Dolmetscher. Allerdings »spricht« ein Browser erst mal nur HTML, eine andere Sprache, in der auch anders programmiert wird als in JavaScript.

Mit der einleitenden Zeile `<script>` wird dem Browser mitgeteilt, dass gleich etwas kommt, das in einer Scriptsprache, wie man JavaScript auch nennt, verfasst ist. Das abschließende `</script>` – mit vorgesetztem Schrägstrich – zeigt dem Browser, dass hier die JavaScript-Anweisungen enden.

Die einzige Zeile in JavaScript

```
document.write("Hallo");
```

bedeutet, dass in das Browserfenster etwas geschrieben werden soll.

Der entsprechende Text wird in Anführungszeichen gesetzt. Das können sowohl einfache wie auch doppelte sein: 'Hallo' ist also ebenso zulässig wie "Hallo".



Du kannst gerne mal das »Hallo« durch einen anderen (längeren) Text ersetzen, um zu sehen, wie das im Browser aussieht.

VISUAL STUDIO CODE STARTEN

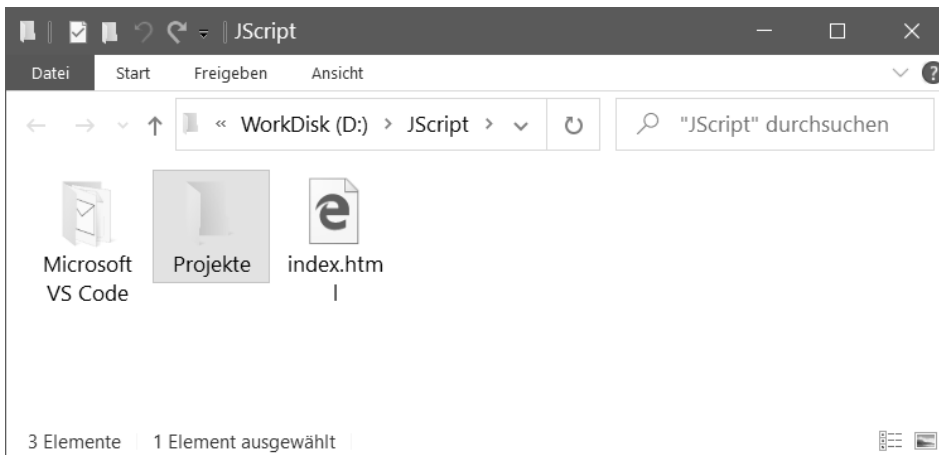
Während der einfache Texteditor am Anfang ausreichen mag, wird er zunehmend unbequem, wenn wir größere Projekte gestalten wollen. Deshalb steigen wir an dieser Stelle um auf einen sogenannten Quelltext-Editor. Ich habe mich für **Visual Studio Code** von Microsoft entschieden (mit dem man übrigens außer in JavaScript noch in vielen anderen Sprachen programmieren kann).

Bevor wir aber damit arbeiten können, muss Visual Studio Code erst installiert werden. Genaueres erfährst du im Anhang A. Hier musst du dir von jemandem helfen lassen, wenn du dir das Einrichten nicht allein zutraust.

Im Folgenden machen wir es uns einfacher und sprechen kurz von VS Code.

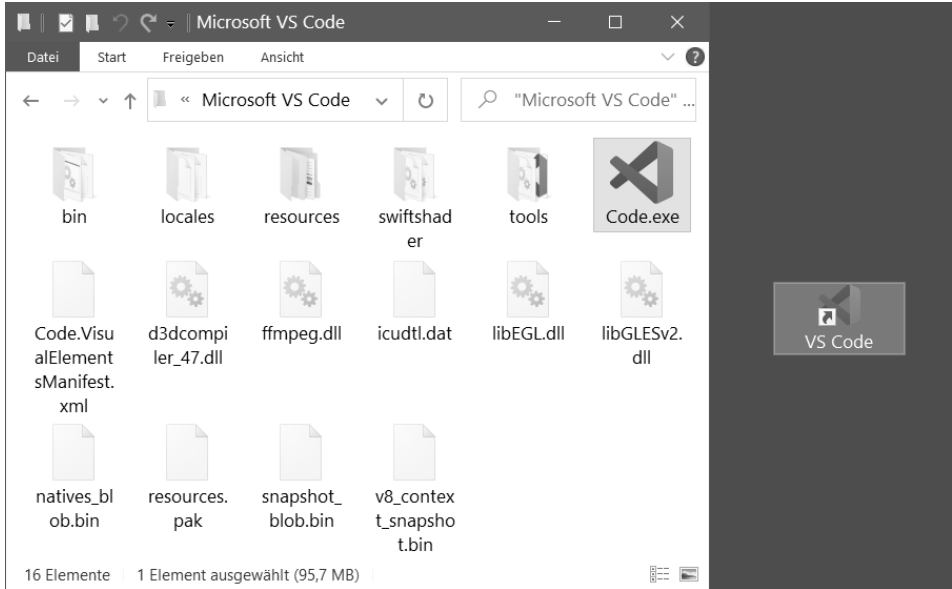


Das Allererste aber, was erledigt werden muss, ist das Erzeugen eines Ordners, in dem alle unsere JavaScript-Projekte gespeichert werden sollen. Ich schlage vor, ihn einfach PROJEKTE zu nennen.



Nun können wir VS Code starten. Eine Möglichkeit ist diese:

- Doppelklicke auf das Desktop-Symbol mit dem Namen VS CODE. Wenn es nicht vorhanden ist, öffne den Ordner, in dem du VS Code untergebracht hast, und doppelklicke mit der Maus auf das Symbol mit dem Namen CODE.EXE.



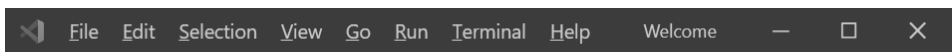
Ich empfehle dir, eine **Verknüpfung** auf dem Desktop anzulegen:

- ❖ Dazu klickst du mit der rechten Maustaste auf das Symbol für VS Code. Im Kontextmenü wählst du **KOPIEREN**.
- ❖ Dann klicke auf eine freie Stelle auf dem Desktop, ebenfalls mit der rechten Maustaste. Im Kontextmenü wählst du **VERKNÜPFUNG EINFÜGEN**.
- ❖ Es ist sinnvoll, für das neue Symbol auf dem Desktop den Text **CODE.EXE – VERKNÜPFUNG** durch **VS CODE** zu ersetzen.

Von nun an kannst du auf das neue Symbol **doppelklicken** und damit VS Code starten.



Je nach Computer kann es eine Weile dauern, bis VS Code geladen ist. Einige Zeit später landest du in einem Willkommen-Fenster.



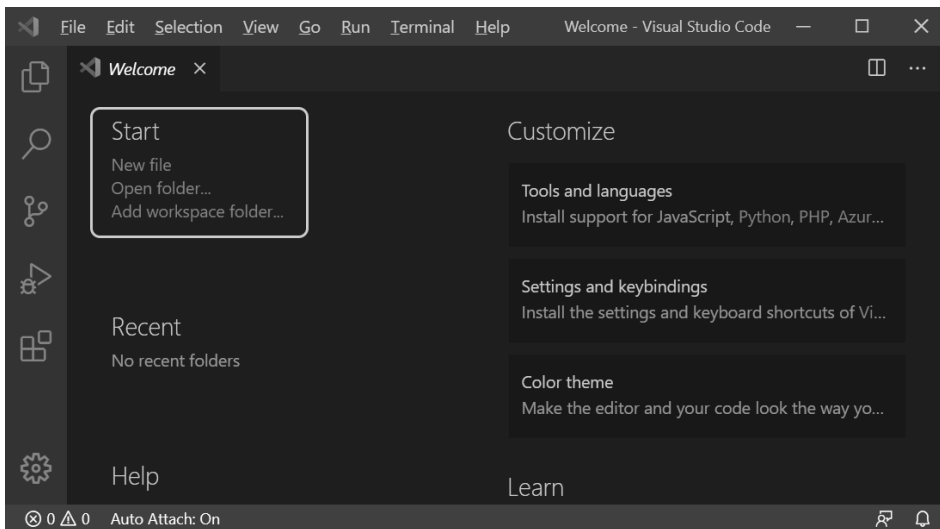
Ganz oben kann man die Menüleiste von Seite 22 erkennen. Von den Menüs wirst du wahrscheinlich diese vier am meisten benutzen:

- ❖ Über das FILE-Menü kannst du Dateien speichern, laden (öffnen), ausdrucken, neu erstellen oder VS Code beenden.
- ❖ Die Menüs EDIT und SELECTION helfen dir bei der Bearbeitung deines Programmtextes, aber auch bei anderen Programmelementen. Außerdem kannst du dort bestimmte Arbeitsschritte rückgängig machen oder wiederherstellen.
- ❖ Über das RUN-Menü sorgst du dafür, dass dein Projekt ausgeführt wird.
- ❖ Und das HELP-Menü bietet dir vielfältige Hilfsinformationen (vor allem auf Englisch) an.

Einige wichtige Menüeinträge sind in einem sogenannten **Popup**-Menü zusammengefasst. Das heißt so, weil es dort aufklappt, wo du gerade mit der rechten Maustaste hin klickst.

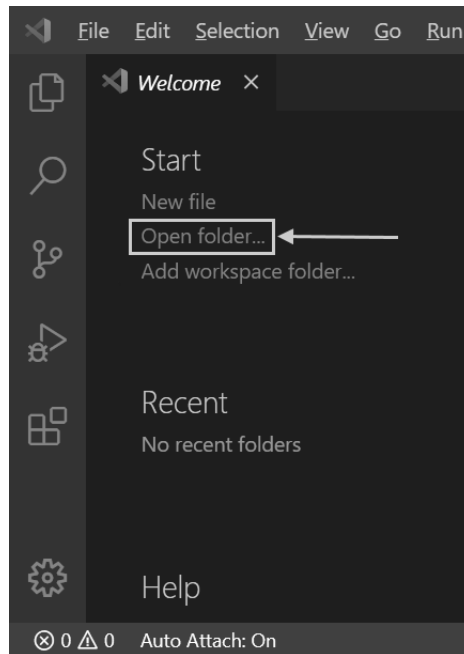


Und nun schauen wir uns mal unter der Menüleiste um. Lass dich durch den Inhalt dieses Fensters nicht verwirren:

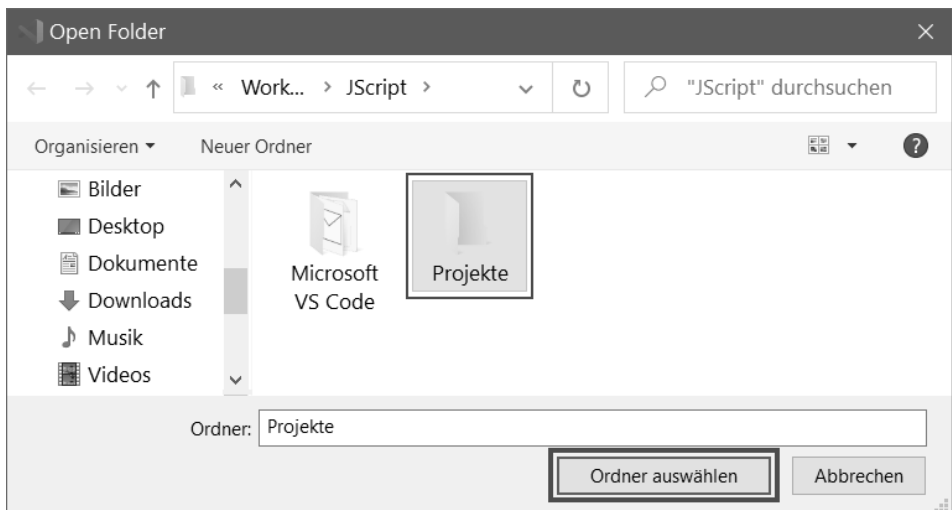


Uns interessiert jetzt das, was (links oben) unter START steht. Bevor wir eine neue Datei erstellen, will VS Code wissen, welchen Ordner wir für unser Programmprojekt ausgewählt haben.

➤ Klicke also auf OPEN FOLDER.

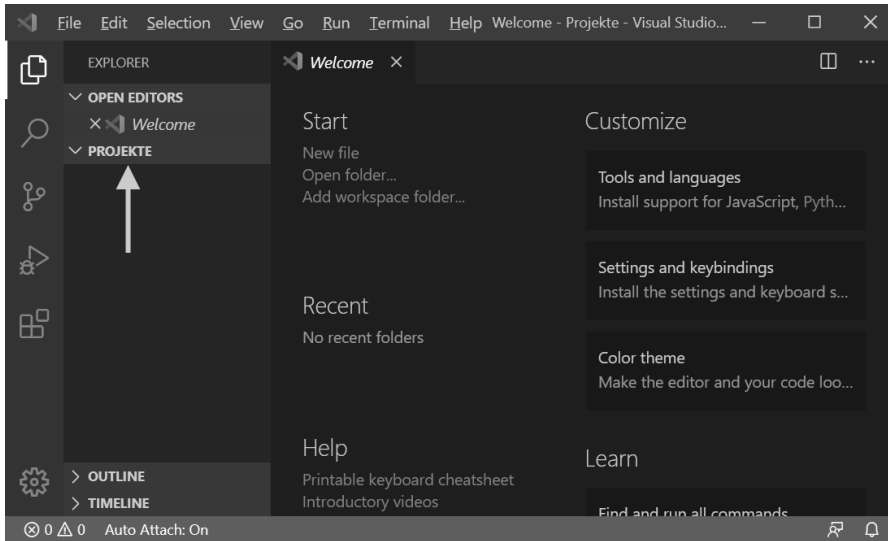


- Im Dialogfenster suchst du nun den Unterordner PROJEKTE (den du vorher erstellt hast).



- Klicke abschließend auf **ORDNER AUSWÄHLEN**.

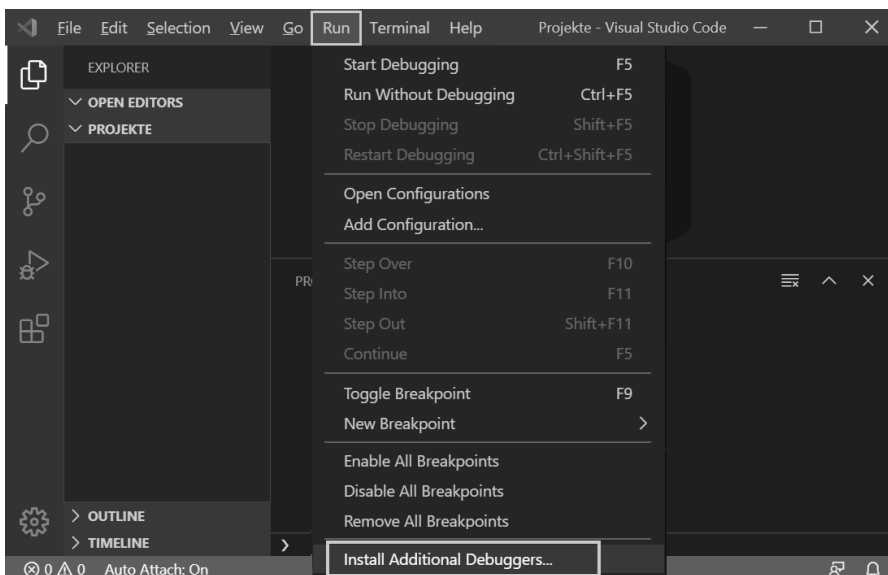
Das Aussehen des Fensters von VS Code hat sich nun ein wenig geändert:



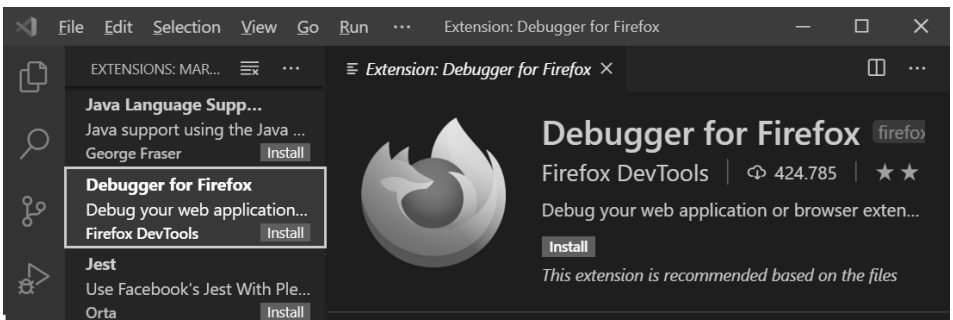
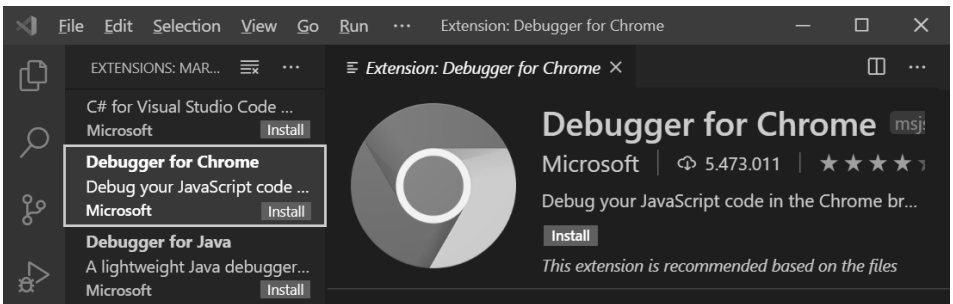
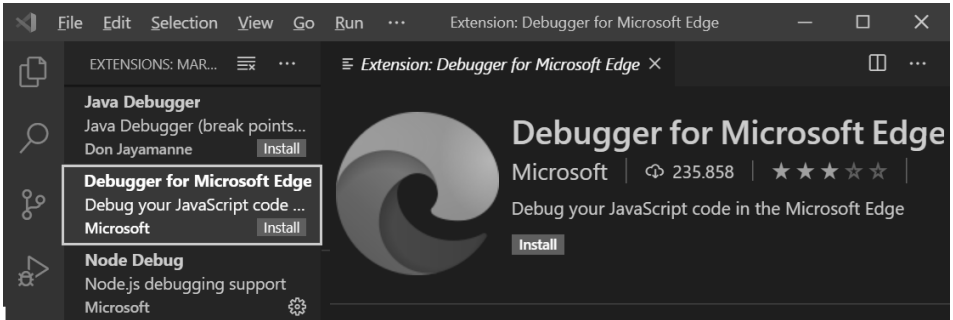
DIE NÖTIGEN ERWEITERUNGEN

Bevor wir jetzt eine neue Datei erzeugen, benötigen wir das passende Hilfsmittel, damit wir später von VS Code aus direkt den Browser starten können, in dem wir das JavaScript-Programm ausführen wollen.

- Klicke dazu oben in der Menüleiste auf RUN und im sich öffnenden Menü ganz unten auf INSTALL ADDITIONAL DEBUGGERS.



Es gibt da ein reichhaltiges Angebot. Für uns interessant sind nur die Erweiterungen für einen der Browser, die wir benutzen, das sind meistens Microsoft Edge, Google Chrome und Mozilla Firefox.

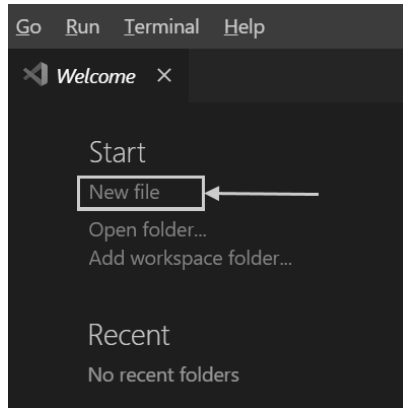


➤ Blättere dich durch das Angebot und suche den Browser aus, den du verwenden willst. Dann klicke dort auf INSTALL.

Kurze Zeit später ist VS Code bereit, deine späteren Programme im entsprechenden Browser zu starten.

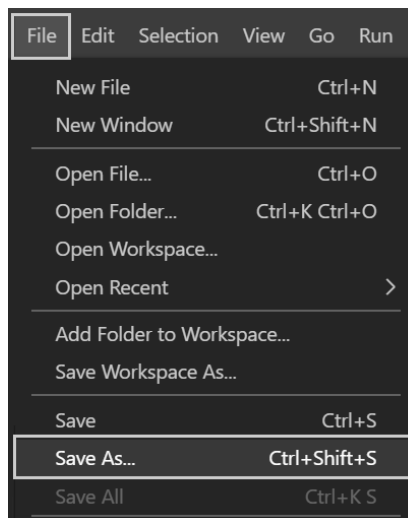
Und nun fangen wir endlich an mit unserem Projekt.

➤ Klicke unter Start auf NEW FILE.

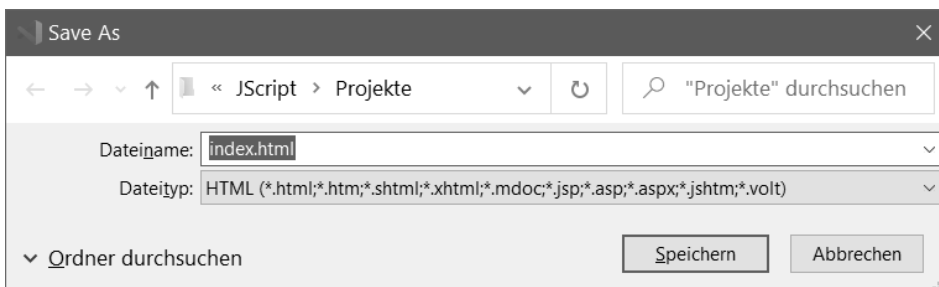


Die neue Datei sollte nun gleich unter dem passenden Namen gespeichert werden, damit VS Code weiß, mit welcher Art von Sprache wir arbeiten wollen.

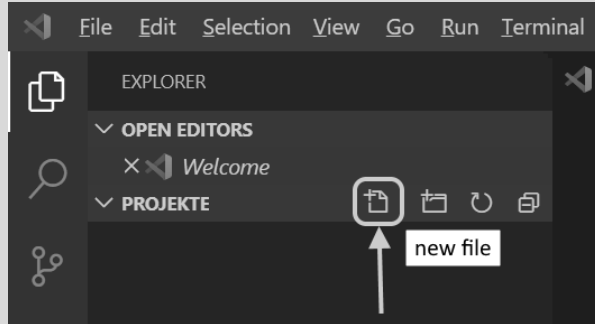
➤ Klicke im Menü auf FILE und dann auf SAVE AS.



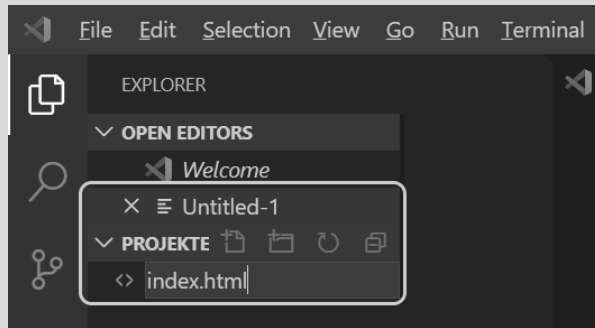
➤ Gib den Namen `index.html` ein und klicke dann auf SPEICHERN.



Alternativ kannst du auch ganz links (im EXPLORER) neben PROJEKTE auf das erste Symbol klicken.



Hier lässt sich dann auch direkt ein neuer Name eingeben. Den Typ erkennt VS Code an der Endung.

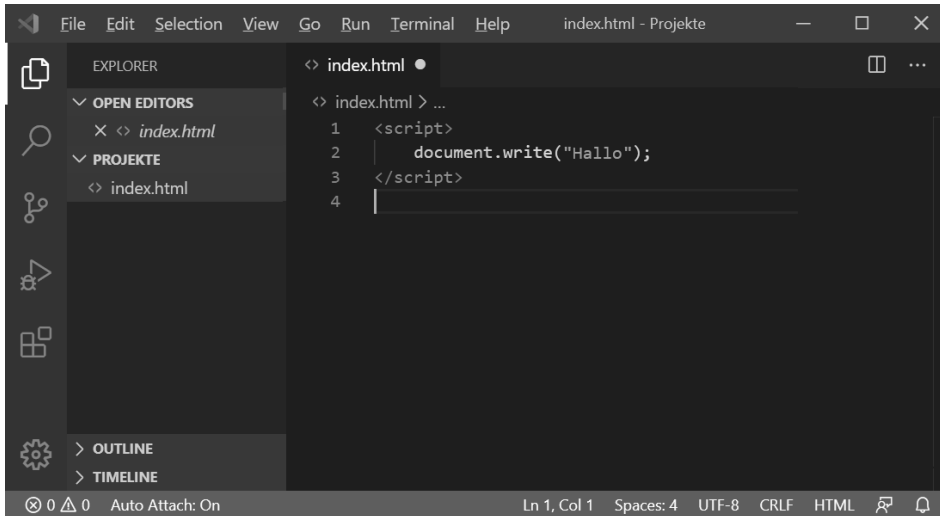


DER STEINIGE WEG ZUM ZWEITEN HALLO

Und nun können wir unseren Programmtext eingeben (auch Quelltext genannt).

➤ Tippe am besten die gleichen Zeilen ein, die wir ganz zu Anfang verwendet haben.

```
<script>
  document.write("Hallo");
</script>
```

```
1 <script>
2   document.write("Hallo");
3 </script>
4
```

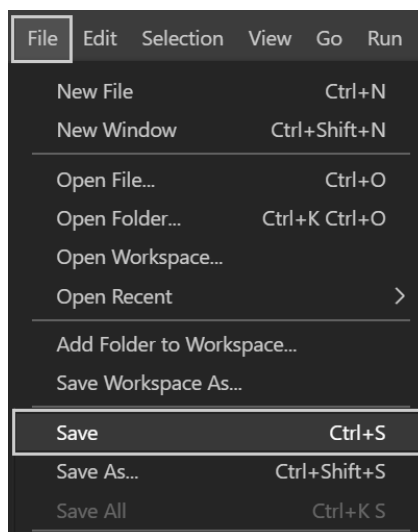
Wie du siehst, wird dein Text schön bunt dargestellt, und die mittlere Zeile ist eingerückt. Sieht doch schick aus! Doch darum geht es nicht, sondern der Quelltext (der ja bei großen Programmen sehr lang werden kann) soll möglichst gut lesbar sein. Unter anderem, damit du Fehler schneller findest.

Klein, aber fein: Wie du siehst, steht am Ende der `write`-Zeile ein Semikolon (;). Das ist wichtig, um jede Anweisung eindeutig abzuschließen.

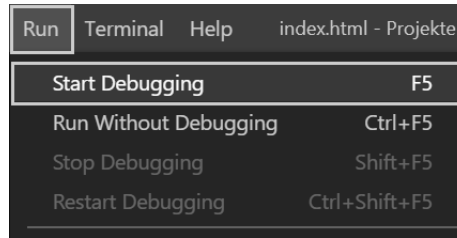


Und jetzt sollten wir das Ganze speichern und dann starten.

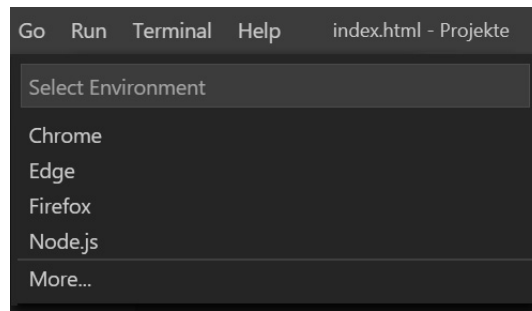
➤ Klicke auf FILE und SAVE oder benutze die Tastenkombination `[Strg] + [S]`.



- Dann klicke auf RUN und auf START DEBUGGING. Du kannst auch die Taste F5 benutzen.



Nun musst du nur noch das »Environment« auswählen, womit die Erweiterung gemeint ist, die dafür sorgt, dass der passende Browser bedient wird.



- Klicke auf einen Eintrag deiner Wahl.

Doch es wird kein Browser gestartet. Stattdessen erscheint auf einmal fast erdrückend viel Text auf der Bildfläche.

```

<> index.html  {} launch.json ×
.vscode > {} launch.json > ...
1  [
2  // Use IntelliSense to learn about possible attributes.
3  // Hover to view descriptions of existing attributes.
4  // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5  "version": "0.2.0",
6  "configurations": [
7    {
8      "type": "edge",
9      "request": "launch",
10     "name": "Edge mit \"localhost\" starten",
11     "url": "http://localhost:8080",
12     "webRoot": "${workspaceFolder}"
13   }
14 ]
15 ]

```

Das ist der Inhalt der Datei LAUNCH.JSON. Sieht verwirrend aus, ist aber nötig, damit VS Code dein Projekt verwalten kann. Denn dort stehen einige Hinweise, die für den Programmstart über den Browser wichtig sind. Diese Datei wird von VS Code automatisch erzeugt.

Wer es genauer wissen will, hier sind ein paar Einzelheiten mehr:

"type" bezeichnet den Browser-Typ, da kann außer "edge" also auch z.B. "chrome" oder "firefox" stehen.

Mit "request": "launch" ist eine Anfrage für den Programmstart gemeint.

"name" steht für den Namen oder eine Erläuterung zur aktuellen Datei.

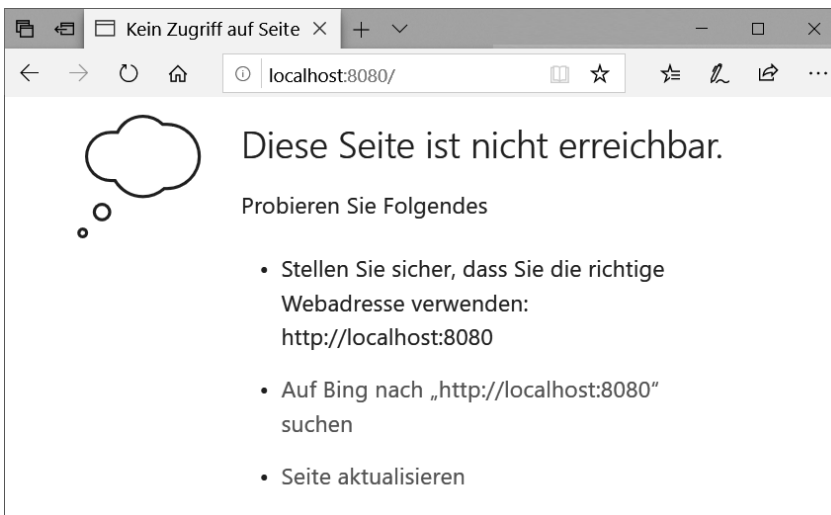
Hinter "url" steht die Adresse, an der der Browser die HTML-Datei finden soll. Liegt die z.B. auf einer Computer-Festplatte, dann setzt man "file:/// " vor den vollständigen Datei-Pfad.

Die Zeile mit "webRoot" kann hier auch weggelassen werden. Außerdem könnte man die oberen drei (grün gefärbten) Zeilen löschen, das sind nur Kommentare.



➤ Lass erst mal alles so, wie es ist, und versuche dann einen Neustart über RUN und START DEBUGGING.

Bei mir führte das zu diesem Ergebnis:



Was ist da los? Nichts anderes, als dass der Browser die angegebene Adresse nicht findet. Er versucht, eine mögliche Seite von dem Computer abzurufen, findet sie aber nicht. In diesem Fall müssen wir direkt den Pfad verwenden, an dem INDEX.HTML zu finden ist. Bei mir liegt sie auf Laufwerk D: im Ordner JSCRIPT und dort im Unterordner PROJEKTE.

Statt

```
http://localhost:8080
```

muss es bei mir heißen:

```
file:///D:/JScript/Projekte/index.html
```

file statt http heißt, dass es sich hier um eine Datei auf einem Datenträger des eigenen Computers handelt. Beachte, dass hinter file 3 (!) Schrägstriche stehen müssen. Solltest du ein anderes Laufwerk oder andere Ordner benutzen, dann musst du die Pfadangabe entsprechend anpassen.

Bei mir sieht die Datei LAUNCH.JSON nun so aus:

```

<> index.html  {} launch.json •
.vscode > {} launch.json > ...
1  {
2    // Use IntelliSense to learn about possible attributes.
3    // Hover to view descriptions of existing attributes.
4    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5    "version": "0.2.0",
6    "configurations": [
7      {
8        "type": "edge",
9        "request": "launch",
10       "name": "Edge mit index.html starten",
11       "url": "file:///D:/JScript/Projekte/index.html"
12     }
13   ]
14 }
15 }

```

Dabei habe ich auch noch den Text hinter "name" angepasst.

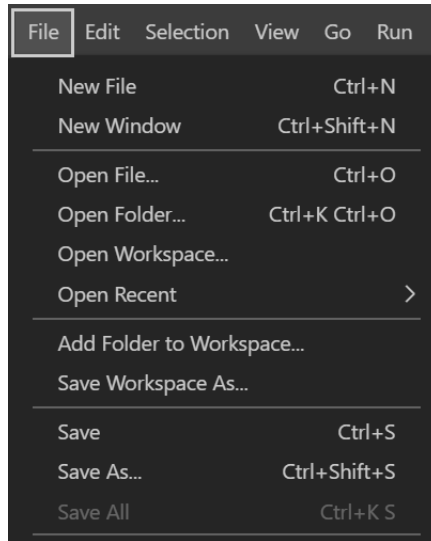
- Ändere die url-Zeile in LAUNCH.JSON entsprechend, damit deine INDEX-Datei gefunden werden kann.
- Wenn du willst, kannst du diese Datei speichern (musst du aber nicht, denn VS Code macht das üblicherweise automatisch).

VISUAL STUDIO CODE BEENDEN

Wir machen jetzt an dieser Stelle eine Verschnaufpause. Eigentlich ist alles so weit fertig für einen Programmlauf, aber den heben wir uns fürs nächste Kapitel auf. Hier sollst du erst noch erfahren, wie man die VS-Code-Umgebung verlässt und beendet. Dazu gibt es nicht nur einen Weg:

ZUSAMMENFASSUNG

- Klicke auf FILE und ganz unten auf EXIT. Oder du klickst auf das kleine X oben rechts in der Titelleiste.



Solltest du vorher noch etwas am Text geändert haben, dann wird das automatisch gespeichert.

Das betrifft sowohl Änderungen in INDEX.HTML als auch in LAUNCH.JSON. Allerdings kannst du beide Dateien während der Bearbeitung jederzeit über FILE und SAVE oder SAVE AS speichern, und das auch beliebig oft.



ZUSAMMENFASSUNG

Nun hast du dein erstes Programm-Projekt fast geschafft. Mal sehen, was du von diesem Kapitel behalten hast. Da wären zuerst mal ein paar Optionen im Umgang mit VS Code:

VS Code starten	Doppelklicke auf das Symbol für VS Code.
Projekt-Ordner öffnen	Klicke auf den Eintrag OPEN FOLDER
Neue Datei erzeugen	Klicke auf den Eintrag NEW FILE
Dateien speichern	Klicke auf FILE/SAVE

Dateien neu speichern	Klicke auf FILE/SAVE AS
Programmprojekt starten	Klicke auf RUN/START DEBUGGING
Hilfesystem aufrufen	Klicke auf HELP oder drücke <code>F1</code>
VS Code beenden	Klicke auf FILE/EXIT

Du kennst die Datei LAUNCH.JSON – ein bisschen. Du weißt, dass ein JavaScript-Programm einen HTML-Rahmen braucht, um in einem Browser zu laufen. Dazu kennst du ein paar Wortschatzbrocken, zum einen von HTML:

<code><script></code>	Anfangs-Marke für ein JavaScript-Programm
<code></script></code>	End-Marke für ein JavaScript-Programm

Zum anderen von JavaScript:

<code>document</code>	Inhalt des Browserfensters
<code>write()</code>	Methode für die Anzeige von Zahlen und Text

EIN PAAR FRAGEN ...

1. Muss ein Programm immer mit `<script>` und `</script>` eingerahmt werden?
2. Wofür ist die Datei LAUNCH.JSON wichtig?

... ABER NOCH KEINE AUFGABE

STICHWORTVERZEICHNIS

-- 87
! 92
{ } 54
*= 94
// 50
/= 94
&& 65
138
++ 84
+= 94
-= 94
|| 65

A

addEventListener 213, 261
Addition 50
alert 67
align 166
Anweisungsblock 46
arc 204
Array 168, 246, 267
 leer 90
 sort 112
Attribut 115
Auflösung 200
Ausschneiden 64
Ausweichen 316

B

Bedingung 45
Beenden
 Programm 73
beginPath 206
Bilderwechsel 315
Bildpunkt 200
<body> 135
body 135
Bogenmaß 209
Boolesche Variable 92

border 220

 78
break 61, 73
Browser 18
 Debugger 25
 Wahl 30
bubbleSort 109
<button> 137
button 137
buttonClick 137

C

Canvas 202, 216
case 60
center 166
charset 63, 134
checkbox 161
checked 159
Chrome 26
class 115
clearInterval 225, 260
clearRect 223
closePath 207
Combobox 156
const 168
constructor 116
continue 74
controlBorder 262
controlContact 270

D

Datei
 finden 32
 Neu 26, 58
 speichern 27
Debugger
 Browser 25
default 61
Dezimalstellen 67

Dezimalzahlen 52
 Division 50
 doctype 132
 document
 getElementByld 144
 write 20
 Document Object Model 132
 Dodging 317
 DOM 132
 Doppelkreuz 138
 Doppelpunkt 61
 do-while 88
 drawImage 221

E

Edge 26
 Editor 12
 Einfügen 64
 else 46
 Endlos-Schleife 86, 87
 Entwicklungsumgebung 12
 Ereignis 141
 Error 37, 40, 44
 Erweiterung 25
 evaluate 277
 Event 141
 event 214, 280
 event.key 281
 Event-Handler 213
 Eventlistener 261
 extends 119
 Extension 25

F

Fallunterscheidung 61
 false 92
 Farbpalette 206
 Fehler 37, 40, 44, 167
 fieldset 159, 166
 file 173
 FileReader 175
 fill 211
 fillRect 211
 fillStyle 210
 fillText 210
 Firefox 26
 fixed 139

floor 67
 font 210
 font-size 244
 for 81, 109, 157
 forEach 301
 function 99
 Funktionskopf 99
 Funktionsrumpf 99

G

Galgenraten 244
 getContext 203
 getElementByld 144, 155
 Globale Variable 102
 Grafik
 Canvas 202
 Farbe 205
 Hintergrund 211
 Kreis 204
 Linie 203
 Pfad 206
 Rechteck 204, 208
 Text 210
 Grafikkarte 200
 Grafik-Pfad 207
 Grenzen 261
 Grundrechenarten 50

H

Hang-Man 244
 <head> 135
 head 135
 height 138, 203
 hidelImage 223
 HTML
 Bedeutung 20
 <html> 133

I

ID 139
 if 42, 44
 img 220
 index 20
 index.html 27
 Infinity 54
 initGame 100

Initialisierung
 Startwerte 100
innerHTML 144
input 146, 157, 173
Installation 323
Instanz 114
Interpreter 11, 20
isHitting 297
isTouching 305
item 108

J

JavaScript 11

K

Kacheln 291
Kapselung 114
key 281
keydown 281, 296
keyup 281
Klammern
 eckig 90, 168
 geschweift 55, 61
 rund 99
 spitz 20
Klasse 114
 Datei 123
Kollisionskontrolle 262, 269, 297
Komma 53
Kommentar 50
Komponente 132
Konstante 168
Konstruktor 116
Kontakt 270
Kontrollfeld 160
Kontrollstruktur 46, 47, 61, 69
Kopieren 64

L

<label> 142
label 142
launch.json 31
Leerzeichen 157
Leerzeile 80
left 138
length 94, 176, 300

let 38
lineTo 203
Listbox 153
load 221
Lokale Variable 101
loopImage 233

M

Malwerkzeuge 203
Math
 floor 67
 PI 209
 random 67
 round 67
Mausklick 261
Mehrfachauswahl 192
Methode 115
mousedown 212
mousemove 212, 280
mouseup 212
moveImage 223
moveTo 203
Multiplikation 50

N

name 158
new 116
New file 26
Notepad 18
Number 52

O

Objektorientierte Programmierung 114
Oder-Operator 65
offsetX 214
offsetY 214
onchange 174
onclick 137, 261
onerror 177
onkeydown 281, 296
onkeyup 281
onload 175, 221, 259
onmousedown 215
onmousemove 215, 280
onmouseup 215
Open folder 23

Operator

- 87
- ! 92
- != 47
- && 65, 263
- ++ 84
- == 44
- || 65, 263
- gleich 65
- größer 65
- kleiner 65
- Rechnen 50
- ungleich 65
- Zuweisung 39
- option 153
- Optionsfeld 157, 190

P

- <p> 134
- padding 166
- paint.net 260
- Palindrom 93
- Parameter 105, 300
- Pfad (Grafik) 207
- Pfeiltasten 296
- PI 209
- Pixel 200
- playGame 100
- pop 300
- Popup 23
- position 139
- Programm
 - Ausgabe 20, 36
 - Eingabe 37
 - Konfiguration 31
 - Start 30
- Programmieren 12
- Programmiersprache 13
- Programmierung
 - objektorientiert 114
- Programm-Modul 123
- Programmordner 23
- Projekt-Ordner 43
- prompt 38, 60
- Punkt 53
- push 300
- px 138

Q

- Quelltext 29
- Quiz 165, 181

R

- radio 157
- Radiobutton 157
- random 67
- Raute 138
- readAsText 175
- readFile 174
- rect 204
- Referenz 111
- Referenz-Übergabe 107
- result 175
- return 120, 276, 297
- round 67
- Rückgabewert 120

S

- Schalt-Variable 92, 186, 213
- Schere – Stein – Papier 240
- Schleife 68, 88
- Schlüsselwörter 56, 97
- Schrittweite 82, 223, 262
- scr 220
- script 19
- select 153
- selectedIndex 155
- Semikolon 29, 41
- setInterval 225, 260
- setState 315
- setTimeout 225
- shift 300
- shiftImage 225, 260
- showImage 222
- size 153
- solid 220
- sort 112
- Sortieren 109
- split 176
- src 124
- strict-Modus 41
- String 51
 - length 94
- stroke 203

strokeStyle 206
strokeText 210
<style> 138
style 138, 220
substr 252
Subtraktion 50
switch 60, 297
switch-Struktur 61

T

Tags 133
Tauschen 104
Text
 verknüpfen 38
text-align 166
this 117
Tile 293
Timer 225
<title> 134
title 134
top 138
Transparenz 259
trim 196
true 92
turnImage 232
type 146
Typumwandlung 52

U

Umlaute 63
Und-Operator 65
unshift 300
Ursprung 200
use strict 41

V

value 146
var 38
Variable 38
 global 102
 lokal 101
Vererbung 119
Vergleichsoperator 44, 65
Verknüpfung 22, 139
Verknüpfungsoperator 65
Virtuell 123

Visual Studio Code 21
VS Code
 beenden 32
 installieren 323
 Menüs 23
 starten 21
 Startfenster 23
 Verknüpfung 22

W

Wert-Übergabe 107
while 68, 84
width 138, 203
Wiederholung 68
window
 alert 67
 prompt 38, 60
Wortschatz 97
write 20
writeln() 79
Würfelspiel 237

X

x-Achse 200

Y

y-Achse 201

Z

z-Achse 201
Zählschleife 81
Zeilentrennung 83
Zeilenvorschub 176
Zuweisung 72
Zuweisungsoperator 39, 44
Zwischenablage 64