

KOSTENLOSE LESEPROBE

JavaScript

von Null bis Praxis

Selbst programmieren. Mit KI üben.

Der Einstieg bis Arrays und Schleifen

24 vollständige Lektionen mit Übungen

Ein eigener Abschlusscheck mit Lösung

```
const ziel = "selbst schreiben";  
const schritte = ["lesen", "üben", "prüfen"];  
for (const schritt of schritte) {  
  console.log(schritt);  
}
```

Johann Eitlhuber

Auszug aus dem Grundlagenband mit 104 Lektionen

Leseprobenfassung · 6. Oktober 2026

Dein Einstieg beginnt hier

Diese kostenlose Leseprobe enthält die Vorbereitung und die ersten 24 Lektionen des Buches. Du beginnst bei den Grundlagen und lernst, kleine Programme mit Bedingungen, Arrays und Schleifen selbst zu schreiben.

So nutzt du die Leseprobe

1. Richte deinen Lernplatz nach der Anleitung ein. Du brauchst einen Computer mit Tastatur; am Handy kannst du lesen, aber der Kurs ist zum Schreiben am Computer gedacht.
2. Lies eine Lektion, tippe das Beispiel selbst und führe es aus. Versuche die Aufgabe zuerst ohne Musterlösung.
3. Kopiere den gemeinsamen Lernprompt und danach den Prompt der jeweiligen Lektion in einen geeigneten KI-Assistenten. Bitte zunächst um Hinweise.
4. Notiere, was du selbst geschrieben hast und wobei du Hilfe gebraucht hast. Wiederhole bei Bedarf, bevor du weitergehst.

Digital lesen und Lösungen prüfen

Die Musterlösungen sind in dieser PDF normal ausgerichtet. Im gedruckten Buch stehen sie auf dem Kopf, damit du sie erst nach deinem Versuch bewusst ansiehst. Vergrößere die PDF bei Bedarf. Die Texte und Lernprompts lassen sich markieren und kopieren; die ursprünglichen Musterlösungen sind Abbildungen.

Der Abschlusscheck beginnt auf Seite 54, seine Musterlösung auf Seite 56. Ein richtiger Versuch mit Hinweisen ist ein Lernschritt; selbstständiges Können zeigt sich auch bei einer veränderten Aufgabe.

Inhalt der Leseprobe

Vorwort	5
Lernmethode und Einführung	6
Vorbereitung: KI, Windows und Mac	8
Dein Lernsystem	16

Teil A · Die ersten Schritte

01 Was ist Programmcode	17
02 Das erste JavaScript-Programm	18
03 Werte und Datentypen	20
04 Variablen mit const	21
05 Veränderliche Werte mit let	23
06 Rechnen mit JavaScript	24
07 Reihenfolge und Klammern	26
08 Vergleiche	27

Teil B · Entscheidungen und Logik

09 if: Code bei einer Bedingung ausführen	30
10 else: der alternative Weg	31
11 else if: mehrere Fälle unterscheiden	32
12 &&: Bedingungen mit UND verbinden	34
13 : Bedingungen mit ODER verbinden	35
14 !: einen Wahrheitswert umkehren	37
15 Verschachtelte Bedingungen	38
16 Bedingungen sinnvoll planen	40
Zwischenstand 1: Grundlogik	42

Teil C · Arrays und Schleifen

17 Arrays: mehrere Werte unter einem Namen	43
18 Array-Index: einzelne Werte finden	44
19 .length: die Länge eines Arrays	45
20 for...of: jedes Element durchlaufen	46
21 Mit Schleifen zählen und summieren	48
22 Durchschnitt berechnen	49
23 Die klassische for-Schleife	50
24 while, break und continue	52

Dein Abschlusscheck	54
KI-Mischtraining und Selbsteinschätzung	55
Musterlösung zum Abschlusscheck	56
So geht dein Lernweg weiter	57

Impressum

Erste Ausgabe 2026

ISBN (Taschenbuch): 9798160345604

Autor und Herausgeber

Johann Eitlhuber

Weiherweg 13

96199 Zapfendorf

Deutschland

Kontakt: Joale689@gmail.com

© 2026 Johann Eitlhuber

Die Codebeispiele dürfen zum Lernen verwendet, verändert und in eigene Programme übernommen werden. Diese Erlaubnis umfasst die Beispiele dieses Buches; für fremde Bibliotheken und Dienste gelten deren eigene Bedingungen. Die übrigen Texte und Abbildungen dürfen nur im Rahmen der gesetzlichen Erlaubnisse oder mit Zustimmung des Rechteinhabers vervielfältigt werden.

Entstehung des Buches

Dieses Buch entstand in Zusammenarbeit mit künstlicher Intelligenz. KI wurde zum Entwerfen von Texten und Beispielen sowie zur Überarbeitung eingesetzt. Die Beispiele wurden im Rahmen der Bearbeitung soweit möglich in ihrer jeweiligen Umgebung geprüft. Dennoch können Fehler verbleiben. Hinweise auf Unstimmigkeiten sind über die Kontaktadresse willkommen.

Hinweise zu Beispielen und Werkzeugen

Die Programme sind bewusst überschaubare Lernbeispiele. Vor dem Einsatz in einer echten Anwendung sind je nach Zweck weitere Prüfungen nötig, etwa zu Sicherheit, Datenschutz und Barrierefreiheit. Software und Online-Dienste entwickeln sich weiter; Bezeichnungen und Bedienoberflächen können sich ändern.

Die genannten Produktnamen gehören ihren jeweiligen Rechteinhabern. Ihre Nennung beschreibt Werkzeuge und begründet keine Zusammenarbeit mit deren Anbietern. Dieses Buch ist eine unabhängige Veröffentlichung.

Vorwort

Ich komme nicht aus der klassischen Softwareentwicklung. Mein Berufsweg begann mit einer handwerklichen Ausbildung; Computer und Programmieren waren zunächst private Interessen. Über eigene Projekte und die visuellen Blueprints von Unreal Engine beschäftigte ich mich nach und nach mit Programmabläufen. Dort konnte ich Zusammenhänge sehen und ausprobieren, ohne bereits eine textbasierte Programmiersprache zu beherrschen.

Später kam künstliche Intelligenz dazu. Mit ihrer Hilfe konnte ich Code lesen, verändern und Projekte voranbringen. Dabei wurde mir eine Lücke deutlich: Ein Programm zum Laufen zu bringen und es selbstständig schreiben zu können sind zwei unterschiedliche Fähigkeiten. Vor einer leeren Datei fehlte mir häufig die Sicherheit, die einzelnen Schritte selbst aufzubauen.

Aus dieser Erfahrung entstand die Idee zu diesem Buch. Ich wollte einen Lernweg, der klein beginnt und immer wieder zum eigenen Schreiben zurückführt. Die Erfahrungen mit visuellen Werkzeugen waren dabei eine Grundlage. Um Programme unabhängig weiterentwickeln zu können, wollte ich auch die Sprache hinter dem Code verstehen.

JavaScript eignet sich für diesen Einstieg, weil sich schon mit wenigen Zeilen sichtbare Ergebnisse erzeugen lassen. Zu Beginn genügt eine kleine Datei. Später kommen Daten, Funktionen, Dateien und eine einfache Oberfläche im Browser hinzu. Welche Werkzeuge du dafür brauchst und wie du sie einrichtest, erklärt die Vorbereitung Schritt für Schritt.

Die KI gehört zur Lernmethode dieses Buches. Sie kann Aufgaben stellen, Fragen beantworten und bei Fehlern Hinweise geben. Damit diese Hilfe zum Lernen beiträgt, soll sie dir Zeit für einen eigenen Versuch lassen. Die fertige Lösung ist der letzte Schritt, wenn du ausdrücklich darum bittest.

Die 104 Lektionen geben dem Lernen eine feste Struktur. Zwei Lektionen pro Woche ergeben einen Weg über 52 Wochen. Das ist eine Orientierung und keine Zusage, nach genau einem Jahr ein bestimmtes Niveau zu erreichen. Manche Themen gehen schnell, andere brauchen Wiederholung. Entscheidend ist, dass du den Code zunehmend selbst erklären, schreiben und verändern kannst.

Dieses Buch vermittelt Grundlagen und erste praktische Anwendungen. Ein zweiter Band ist geplant, der darauf aufbaut und fortgeschrittene Themen sowie größere Projekte behandelt. Auch danach bleiben Dokumentation, Übung und neue Fragen Teil des Programmierens. Mein Ziel ist, dass du eigene Vorhaben in überschaubare Schritte zerlegen und daraus verstandenen Code entwickeln kannst.

Johann Eitlhuber

So benutzt du dieses Buch

Der Kurs richtet sich an Menschen ohne Programmiererfahrung. Du musst keine Befehle auswendig kennen. Für die ersten Aufgaben reichen einfache Rechnungen und die Bereitschaft, regelmäßig selbst zu schreiben. Wenn du auch am Computer noch unsicher bist, nimm dir für die Einrichtung Zeit und lass dir jeden unbekannten Schritt erklären.

Was dieser Band vermittelt

Du lernst Werte, Variablen, Bedingungen, Schleifen, Arrays, Objekte und Funktionen kennen. Darauf bauen erste Übungen mit Dateien, JSON, Schnittstellen und kleinen Browseroberflächen auf. Fehlersuche, einfache Tests und Versionsverwaltung helfen dir, die eigenen Programme nachvollziehbar weiterzuentwickeln.

Der Kurs ist ein breites Fundament. Er behandelt nicht sämtliche Sprachdetails und auch nicht alle Werkzeuge des JavaScript-Umfelds. Große Frameworks, komplexe Anwendungen und vertiefende Architekturfragen gehören nicht zum Lernziel dieses Bandes. Die Abschlussaufgaben zeigen dir, welche Grundlagen du bereits selbstständig anwenden kannst.

Dein Lernrhythmus

Plane zunächst zwei Lektionen pro Woche. Zu einer Lektion gehören Lesen, Schreiben, Ausführen und zusätzliche Übungen. Brauchst du länger, wiederhole das Thema, bevor du weitergehst. Wer schneller vorankommt, kann den Rhythmus anpassen. Die Wochenzahl ist keine Prüfung und kein Wettlauf.

- Lies die Erklärung zuerst in Ruhe und versuche, den Ablauf mit eigenen Worten zu beschreiben.
- Tippe das Beispiel selbst, speichere es und führe es aus. Achte darauf, ob die Ausgabe zu deiner Erwartung passt.
- Beginne jede neue, unabhängige Übung mit einer leeren Übungsdatei. So übst du, Syntax und Struktur selbst aufzubauen. Bei mehrteiligen Projekten behältst du die benötigten Dateien.
- Versuche die Buchaufgabe zuerst selbst. Nachschlagen ist erlaubt. Nutze bei Schwierigkeiten zunächst Hinweise und prüfe die Musterlösung erst danach.
- Bearbeite zusätzliche Aufgaben mit dem Lernprompt der Lektion. Lass ältere Themen wiederholen und miteinander verbinden.

Verstehen und selbst schreiben

Wenn du eine Erklärung nachvollziehen kannst, hast du einen wichtigen Schritt geschafft. Ob du das Gelernte selbst anwenden kannst, zeigt sich vor einer leeren Datei

und bei einer leicht veränderten Aufgabe. Prüfe deshalb regelmäßig beides: Kannst du den Ablauf erklären und kannst du ihn ohne vorgegebene Lösung schreiben?

Die kleinen Buchübungen sind ein erster Test. Für Sicherheit brauchst du weitere Varianten und Wiederholungen. Die KI-Prompts am Ende der Lektionen gehören deshalb zum vorgesehenen Lernweg. Die JavaScript-Programme selbst laufen unabhängig von einer KI; die zusätzliche Übungsbegleitung übernimmt ein geeigneter KI-Assistent.

Die Musterlösungen

Die Musterlösungen sind in dieser digitalen Leseprobe normal ausgerichtet. Schau sie erst nach deinem eigenen Versuch an. Im gedruckten Buch stehen sie auf dem Kopf. Eine andere Umsetzung kann ebenso richtig sein, wenn sie dieselbe Aufgabe erfüllt.

Bevor wir anfangen Was ist JavaScript überhaupt

JavaScript ist eine Programmiersprache. Du schreibst Anweisungen in eine Datei; ein dafür geeignetes Programm führt sie aus. Damit kannst du beispielsweise Preise berechnen, eine Liste auswerten oder auf einen Klick reagieren.

Woher kommt JavaScript

Die erste Version entstand 1995 bei Netscape und wurde von Brendan Eich entwickelt. JavaScript sollte Webseiten interaktiv machen. Die Sprache wurde später standardisiert; ihr Sprachstandard heißt ECMAScript. Java und JavaScript sind trotz der ähnlichen Namen verschiedene Programmiersprachen.

Wofür eignet sich die Sprache

Im Browser steuert JavaScript interaktive Webseiten. Mit Node.js läuft es auch außerhalb des Browsers, etwa für kleine Hilfsprogramme, Dateiverarbeitung oder Serveranwendungen. Dadurch kannst du mit derselben Sprache verschiedene Aufgaben ausprobieren.

Eine Sprache ist vor allem dann passend, wenn sie zu einem Vorhaben und seiner Umgebung passt. JavaScript hat im Browser einen festen Platz und ein großes Angebot an Werkzeugen. Andere Sprachen können für andere Aufgaben günstiger sein, etwa wenn bestimmte Bibliotheken, sehr aufwendige Berechnungen oder besondere Systemanforderungen entscheidend sind. Solche Unterschiede machen JavaScript nicht zu einer minderwertigen Sprache.

Eine Besonderheit wirst du früh kennenlernen: JavaScript kann Werte automatisch umwandeln. Dadurch sind knappe Schreibweisen möglich, aber auch überraschende Ergebnisse. Wir verwenden deshalb klare Datentypen und nachvollziehbare

Prüfungen. Zunächst lernst du kleine Programme mit einer Textausgabe; eine Browseroberfläche folgt später.

Dein Schreibplatz und der Startknopf

In Visual Studio Code, kurz VS Code, schreibst und speicherst du deine Programme. Es ist ein Textprogramm, das auf Code spezialisiert ist. Node.js führt deine JavaScript-Dateien aus. Die Erweiterung Code Runner gibt dir dafür einen bequemen Startknopf in VS Code.

PowerShell ist das Befehlsfenster von Windows. Darin kannst du Programme starten und mit Dateien oder Ordnern arbeiten. Für den Kurs reichen wenige Befehle. Du benutzt es bei der Einrichtung und später wieder für Projekte, Tests und Git. Die KI hilft dir bei Fragen und Übungen; sie ersetzt keines dieser Programme.

Vorbereitung Was du für dieses Buch brauchst

Du brauchst einen Windows-Computer oder einen Mac, eine Tastatur und Internet für Downloads und die KI-Begleitung. Ein Smartphone ist praktisch für Fotos, aber keine Voraussetzung. Die Windows-Einrichtung folgt unten; eine eigene Anleitung für den Mac steht direkt danach. Tablets und Smartphones ersetzen für diesen Kurs keinen Computer.

Installiere Programme ausschließlich von den angegebenen offiziellen Seiten. Die benötigten Werkzeuge können kostenlos verwendet werden. Auch bei der KI kannst du zunächst mit einem kostenlosen Angebot beginnen. Für Konten gelten die jeweiligen Bedingungen und gegebenenfalls Altersgrenzen der Anbieter.

Zuerst die KI Begleitung einrichten

Wähle einen KI-Assistenten, der Gespräche führen, Programmcode erklären und bei Bedarf Bilder lesen kann. Die folgenden Angebote kommen dafür infrage. Das Buch setzt keinen bestimmten Anbieter voraus und enthält keine Empfehlung für ein kostenpflichtiges Abonnement.

ChatGPT von OpenAI: <https://chatgpt.com>

Claude von Anthropic: <https://claude.ai>

Gemini von Google: <https://gemini.google.com>

Microsoft Copilot: <https://copilot.microsoft.com>

Alle vier können bei Erklärungen, Lernaufgaben und Fehlersuche helfen; sie unterstützen auch Bild-Uploads. Wie gut eine Antwort passt, hängt vom Modell, der Frage und den verfügbaren Funktionen ab. Kein Anbieter liefert immer korrekte Antworten. Verlasse dich bei technischen Einzelheiten im Zweifel auf die offizielle Dokumentation und das tatsächlich beobachtete Programmmergebnis.

Öffne die gewählte Webseite im Browser. Ein Browser ist das Programm, mit dem du Internetseiten aufrufst, zum Beispiel Edge, Firefox, Chrome oder Safari. Melde dich an oder lege über die angebotene Registrierung ein Konto an. Nutze deine eigenen Zugangsdaten und folge einer gegebenenfalls angeforderten Bestätigung per E-Mail. Für den Einstieg wählst du ein kostenloses Angebot, soweit es für dein Konto verfügbar ist.

Bei Gemini erfolgt die Anmeldung über ein Google-Konto, bei Copilot über ein Microsoft-Konto. ChatGPT und Claude bieten auf ihren Seiten eigene Anmeldewege an. Prüfe vor einer Bestätigung, ob du ein kostenloses Konto anlegst oder gerade einen kostenpflichtigen Tarif auswählst. Ein Abo ist für den Einstieg nicht eingeplant.

Ein Lernchat für deinen Kurs

Ein Chat ist ein Gespräch mit der KI. Eine Nachricht kann eine Frage, dein Code oder eine Anweisung enthalten. Eine solche Anweisung nennt man Prompt. Eröffne ein Gespräch für deinen JavaScript-Kurs und gib ihm, soweit möglich, einen passenden Namen, etwa JavaScript lernen.

Viele Angebote zeigen bisherige Gespräche in einer seitlichen Liste. Auf einem schmalen Bildschirm findest du sie häufig über ein Menü. Öffne bei der nächsten Sitzung deinen vorhandenen Lernchat. Mit Neues Gespräch beginnst du dagegen wieder von vorn. Die genaue Bezeichnung kann sich je nach Anbieter ändern.

Wenn du zusätzlich dein Smartphone benutzt, öffne dort dieselbe Webseite und melde dich mit demselben Konto an. Ein gespeichertes Gespräch kannst du bei Angeboten mit geräteübergreifendem Verlauf auch dort wieder öffnen. Ein temporärer oder nicht gespeicherter Chat wird gegebenenfalls nicht übernommen. Prüfe, ob das bisherige Gespräch in deiner Liste erscheint.

Kostenlose Nutzung und Gesprächsgrenzen

Kostenlose Angebote haben Nutzungslimits. Es kann separate Grenzen für Nachrichten, stärkere Modelle und Bild-Uploads geben. Die Zahl der erlaubten Anfragen kann sich ändern. Auch beim Üben ist ein Limit möglich. Wenn ein Angebot eine Pause verlangt, kannst du die Buchaufgaben weiterbearbeiten und später in den Chat zurückkehren.

KI verarbeitet Text in kleinen Einheiten, den Tokens. Ein Token ist häufig ein Wortteil, ein kurzes Wort oder ein Satzzeichen. Lange Nachrichten, umfangreicher Code und lange Gespräche benötigen mehr solcher Einheiten. Die KI kann nur eine begrenzte Menge Inhalt gleichzeitig berücksichtigen; auch ein gespeicherter Chat ist kein unbegrenztes Gedächtnis.

Halte deshalb in einer eigenen Notiz fest, welche Lektion du bearbeitet hast und welche Befehle du bereits kennst. Beginne eine neue Sitzung mit deinem aktuellen Lernstand. Wenn ein Gespräch sehr lang wird oder die KI frühere Regeln nicht mehr

beachtet, starte mit dieser Notiz und dem Lernprompt ein neues Gespräch. Notiere keine Passwörter oder vertraulichen Daten im Lernchat.

Der gemeinsame Lernprompt

Kopiere den folgenden Text zu Beginn deines Lernchats. Er legt fest, wie die KI dir helfen soll. Der zusätzliche Prompt einer Lektion nennt danach das aktuelle Thema. Du musst diesen allgemeinen Text nicht bei jeder Aufgabe erneut senden; in einem neuen Chat solltest du ihn wieder mitgeben.

Ich lerne JavaScript von Grund auf mit dem Buch JavaScript von Null bis Praxis. Behandle mich als Anfänger. Arbeite in kleinen Schritten und stelle immer nur eine Aufgabe gleichzeitig. Gib mir keine fertige Lösung, bevor ich ausdrücklich darum bitte. Lass mich bei einem Fehler zuerst selbst suchen. Gib danach zunächst einen kleinen Hinweis, bei Bedarf einen deutlicheren. Nutze nur Themen und Befehle, die ich bereits gelernt habe; frage bei Unsicherheit nach meinem Lernstand. Lass mich ältere Inhalte regelmäßig wiederholen und mit dem neuen Thema kombinieren. Unterscheide zwischen einer Erklärung, die ich verstehe, und Code, den ich selbstständig schreiben kann. Warte nach jeder Aufgabe auf meinen Versuch.

Fotos und Fehlermeldungen verwenden

Wenn du eine Buchstelle nicht verstehst, kannst du sie mit dem Smartphone fotografieren und über die Bild- oder Dateifunktion der KI hochladen. Fotografiere den Text gerade, scharf und bei gutem Licht. Lade möglichst nur den relevanten Abschnitt hoch. Schreibe dazu, welches Wort oder welcher Schritt unklar ist.

Auch ein Bild deiner Fehlermeldung kann helfen. Wenn du den Text kopieren kannst, füge ihn zusätzlich ein: Das erleichtert die genaue Prüfung. Unter Windows erstellt Windows+Umschalt+S einen Bildschirmausschnitt. Öffne ihn anschließend im Snipping Tool und speichere ihn bei Bedarf mit Strg+S als Bilddatei. Merke dir den Speicherort. Auf dem Mac verwendest du Umschalt+Befehl+4; der Ausschnitt wird normalerweise auf dem Desktop gespeichert. Lade die Bilddatei anschließend im Lernchat hoch.

Für einen längeren Einrichtungsbefehl aus dem gedruckten Buch kannst du die KI bitten, den fotografierten Block als kopierbaren Text auszugeben. Vergleiche das Ergebnis vor dem Ausführen mit der Vorlage. Besonders Anführungszeichen, Minuszeichen, Klammern und Zeilenumbrüche können beim Erkennen verwechselt werden.

Schreibe den Befehlsblock auf diesem Foto als kopierbaren Text ab. Verändere keine Befehle und ergänze nichts. Markiere Stellen, die du nicht sicher lesen kannst. Erkläre anschließend kurz, was der Block macht. Ich werde ihn vor dem Ausführen mit dem Buch vergleichen.

Mit der KI durch die Einrichtung

Wenn du die folgenden Schritte mit Begleitung machen möchtest, sende diesen Prompt. Nenne dein tatsächliches Betriebssystem und gib bei Bedarf die passende

Buchstelle oder ein Bildschirmfoto dazu. Führe immer nur den nächsten erklärten Schritt aus.

Ich richte meinen Computer für das Buch JavaScript von Null bis Praxis ein und habe keine Programmiererfahrung. Frage zuerst, ob ich Windows oder macOS benutze und was bereits installiert ist. Ziel: VS Code öffnet den Ordner JavaScript-Training, Node.js funktioniert und Code Runner startet uebung.js. Erkläre immer nur einen nächsten Schritt und warte auf meine Rückmeldung. Verwende offizielle Downloadseiten. Gib Befehle vollständig kopierbar aus und erkläre vor dem Ausführen kurz, was sie bewirken. Gib keine pauschalen Befehle zum Löschen von Dateien oder Abschalten von Schutzfunktionen. Orientiere dich an der Anleitung aus dem Buch.

Windows Schritt 1 Visual Studio Code installieren

Öffne <https://code.visualstudio.com> und lade Visual Studio Code für Windows herunter. Öffne die heruntergeladene Installationsdatei. Die Standardauswahl reicht für den Kurs. Falls angeboten, aktiviere Zu PATH hinzufügen; dadurch wird später der Befehl code gefunden. Visual Studio ist ein anderes Produkt. Für dieses Buch brauchst du Visual Studio Code.

Windows Schritt 2 Node installieren und prüfen

Öffne <https://nodejs.org> und lade die für Windows angebotene LTS-Version herunter. LTS bezeichnet eine länger unterstützte Version. Installiere sie mit den Standardoptionen. Node.js ist das Programm, das deinen JavaScript-Code ausführt. npm wird mitinstalliert und hilft später bei der Verwaltung zusätzlicher Programmpakete.

Öffne danach ein neues Befehlsfenster: Drücke die Windows-Taste, tippe PowerShell und öffne das entsprechende Programm. Bei Windows Terminal sollte ein PowerShell-Tab aktiv sein. Die Zeile vor dem blinkenden Cursor kann mit PS beginnen und einen Ordnerpfad anzeigen. Sie gehört nicht zum Befehl, den du eingibst.

Gib die beiden Befehle einzeln ein und drücke jeweils Enter. Eine Versionsnummer nach jedem Befehl zeigt, dass das Programm gefunden wurde. Die Zahlen können anders aussehen als in einer Anleitung.

```
node -v  
npm.cmd -v
```

Unter Windows verwenden wir npm.cmd. Das startet npm, ohne eine eventuell gesperrte PowerShell-Skriptdatei auszuführen. Wenn ein Befehl nicht gefunden wird, schließe das Fenster und öffne PowerShell erneut. Hilft das nicht, prüfe mit der KI die Installation und die genaue Fehlermeldung; ändere nicht vorschnell Sicherheitseinstellungen.

Windows Schritt 3 Deinen Lernordner öffnen

Gehe auf deinen Desktop. Klicke mit der rechten Maustaste auf eine freie Stelle und wähle Neu > Ordner. Nenne ihn JavaScript-Training. In diesem Ordner liegen deine Übungsdateien; so findest du sie später wieder.

Öffne VS Code über das Startmenü. Wähle Datei > Ordner öffnen und anschließend den neuen Ordner auf dem Desktop. Falls du für diesen selbst erstellten Ordner nach Vertrauen gefragt wirst, bestätige es. Links im Explorer von VS Code sollte nun JavaScript-Training erscheinen. Dieser Explorer zeigt die Dateien deines geöffneten Ordners.

Windows Schritt 4 Den Startknopf installieren

Klicke links in VS Code auf Erweiterungen, das Symbol mit mehreren kleinen Quadraten. Suche nach Code Runner. Wähle die Erweiterung von Jun Han mit der Kennung formulahendry.code-runner und klicke auf Installieren. Code Runner startet später Node.js über eine Schaltfläche; es ist keine zusätzliche Programmiersprache.

Windows Schritt 5 Die Übungsdatei anlegen

Wechsle links zurück zum Explorer. Neben deinem Ordner findest du die Schaltfläche Neue Datei. Klicke darauf, tippe uebung.js und bestätige mit Enter. Alternativ wählst du Datei > Neue Textdatei und speicherst sie als uebung.js in JavaScript-Training. Achte auf die Endung .js; die Datei soll nicht uebung.js.txt heißen.

Windows Schritt 6 Die Ausgabe übersichtlich einstellen

Für diesen neuen Lernordner stellen wir ein, dass Code Runner die Datei vor dem Start speichert und alte Ausgaben leert. Drücke in VS Code Strg+Umschalt+P. Dieses Fenster heißt Befehlspalette. Suche nach Preferences: Open Workspace Settings (JSON) beziehungsweise der deutschen Entsprechung Arbeitsbereichseinstellungen öffnen (JSON). Wähle die JSON-Datei des Arbeitsbereichs, nicht die Benutzereinstellungen.

Im frisch angelegten Lernordner kannst du den Inhalt dieser Einstellungsdatei durch den folgenden Block ersetzen. Speichere mit Strg+S. Der Block ist eine Einstellung für VS Code und gehört nicht in uebung.js. Wenn schon eigene Einstellungen vorhanden sind, lass dir beim Ergänzen helfen, statt sie zu überschreiben.

```
{
  "code-runner.runInTerminal": false,
  "code-runner.showExecutionMessage": false,
  "code-runner.clearPreviousOutput": true,
  "code-runner.saveFileBeforeRun": true,
  "code-runner.ignoreSelection": true,
  "code-runner.executorMap": {
    "javascript": "node"
  }
}
```

VS Code speichert diese Einstellungen für deinen Ordner in .vscode/settings.json. Du musst den Block noch nicht als Programm verstehen. Für die Einrichtung ist Kopieren in Ordnung. Die JavaScript-Beispiele im eigentlichen Kurs tippst du dagegen selbst.

Windows Schritt 7 Hallo Welt ausgeben

Öffne wieder `uebung.js` und tippe diese eine Zeile. Verwende gerade Anführungszeichen, wie im Beispiel, und speichere mit `Strg+S`.

```
console.log("Hallo Welt");
```

Klicke oben rechts auf den Startknopf von Code Runner. Beim Darüberfahren steht dort `Run Code` beziehungsweise `Code ausführen`. Alternativ klicke mit der rechten Maustaste in die Datei und wähle `Run Code`. Im unteren Fenster Ausgabe sollte `Hallo Welt` erscheinen. Falls mehrere Ausgabekanäle angeboten werden, wähle den Kanal `Code`.

`Hallo Welt` ist ein verbreitetes erstes Testprogramm. Es zeigt, dass die Einrichtung funktioniert. Die Anweisung `console.log` erzeugt eine Ausgabe; angezeigt wird hier der Text `Hallo Welt`. Die einzelnen Zeichen der Anweisung erklärt Lektion 2.

Du kannst die Datei auch über das Befehlsfenster starten. Öffne in VS Code Terminal > Neues Terminal. Ist JavaScript-Training der aktuelle Ordner, lautet der Windows-Befehl:

```
node .\uebung.js
```

Das Terminal ist das Befehlsfenster innerhalb von VS Code. Dort kann PowerShell laufen. Ausgabe und Terminal sind zwei unterschiedliche Bereiche: Code Runner zeigt unsere ersten Ergebnisse im Fenster Ausgabe, ein Terminalbefehl zeigt sie im Terminal. Beide starten dasselbe JavaScript-Programm.

Windows Schritt 8 Ein Symbol auf dem Desktop

Mit einer Verknüpfung öffnest du später direkt den Lernordner und `uebung.js`. Prüfe zunächst im PowerShell-Terminal den Befehl `code --version`. Wenn er nicht gefunden wird, starte VS Code und das Terminal neu. Für die folgende Verknüpfung muss der Befehl `code` aus der üblichen Windows-Installation erreichbar sein.

Kopiere den folgenden Block vollständig in PowerShell. Er legt eine Verknüpfung `JavaScript lernen` auf deinem Desktop an. Deine JavaScript-Datei muss bereits im Ordner `JavaScript-Training` liegen. Der Block schreibt nur die Verknüpfung und verändert deinen Übungscode nicht.

```
$jsDesktop = [Environment]::GetFolderPath("Desktop")
$jsOrdner = Join-Path $jsDesktop "JavaScript-Training"
$jsDatei = Join-Path $jsOrdner "uebung.js"
$jsBefehl = Get-Command code -ErrorAction Stop
$jsBin = Split-Path $jsBefehl.Source -Parent
$jsProgramm = Join-Path $jsBin "..\Code.exe"
if (!(Test-Path $jsProgramm)) {
    throw "VS Code nicht gefunden. Anleitung prüfen."
}
$jsShell = New-Object -ComObject WScript.Shell
$jsLinkPfad = Join-Path $jsDesktop "JavaScript lernen.lnk"
$jsLink = $jsShell.CreateShortcut($jsLinkPfad)
$jsLink.TargetPath = $jsProgramm
$jsLink.Arguments = '"' + $jsOrdner + ' ' + $jsDatei + '"'
$jsLink.WorkingDirectory = $jsOrdner
$jsLink.Save()
```

Doppelklicke anschließend auf JavaScript lernen. Prüfe, ob der richtige Ordner und uebung.js geöffnet werden. Wenn VS Code an einem anderen Ort installiert ist oder der Ordner anders heißt, kann die KI anhand der Meldung bei der Anpassung helfen. Bis dahin öffnest du den Lernordner über Datei > Ordner öffnen.

PowerShell im weiteren Kurs

PowerShell kann Programme starten und Dateien oder Ordner bearbeiten. Ein Befehl kann daher auch etwas verändern oder löschen. Im Kurs verwendest du deine eigenen Übungsordner. Führe einen unbekannten Befehl erst aus, wenn du seinen Zweck verstanden hast. Du brauchst für die normalen Übungen kein Befehlsfenster mit Administratorrechten.

Drei einfache Befehle helfen bei der Orientierung. Gib den Teil vor dem Zeichen # ein; dahinter steht im Beispiel eine Erklärung. pwd zeigt deinen aktuellen Ordner, dir dessen Inhalt und cd .. wechselt zum übergeordneten Ordner.

```
pwd          # aktueller Ordner
dir          # Dateien und Unterordner
cd ..       # einen Ordner zurück
```

Einrichtung auf dem Mac

Auf einem Mac lernst du dieselbe Sprache. Die Windows-Befehle zur Einrichtung und die Desktop-Verknüpfung verwendest du dort nicht. Statt PowerShell benutzt du die App Terminal oder das Terminal in VS Code. Öffne Terminal über die Suche des Macs oder den Ordner Programme > Dienstprogramme.

1. Lade VS Code für macOS von <https://code.visualstudio.com> herunter. Öffne das heruntergeladene Archiv, falls nötig, und verschiebe Visual Studio Code in Programme. Starte es dort.

2. Lade die angebotene LTS-Version von Node.js für macOS von <https://nodejs.org> herunter und folge der Installation. Öffne danach ein neues Terminal und prüfe die Installation mit den beiden Befehlen:

```
node -v  
npm -v
```

3. Lege im Finder auf dem Desktop einen Ordner JavaScript-Training an. Öffne ihn in VS Code über Datei > Ordner öffnen. Erstelle darin uebung.js, wie im Windows-Schritt 5 beschrieben.

4. Installiere Code Runner von Jun Han über Erweiterungen. Öffne mit Umschalt+Befehl+P die Befehlspalette, suche die Arbeitsbereichseinstellungen als JSON und übernimm in deinem neuen Ordner den Einstellungsblock aus Windows-Schritt 6. Speichere auf dem Mac mit Befehl+S.

5. Tippe `console.log("Hallo Welt");` in uebung.js. Starte über Run Code und prüfe die Ausgabe. Im Terminal kannst du dieselbe Datei aus ihrem Ordner mit `node ./uebung.js` starten.

Wenn du VS Code über den Befehl code öffnen möchtest, suche in seiner Befehlspalette nach Shell Command: Install code command in PATH und führe den passenden Eintrag aus. Öffne danach das Terminal erneut. Über Datei > Ordner öffnen kannst du den Lernordner jederzeit auch ohne diesen Befehl öffnen.

Windows Befehle auf den Mac übertragen

Die JavaScript-Beispiele gelten auf beiden Systemen. Bei Startbefehlen und Tastenkürzeln unterscheiden sich einige Angaben. Die folgenden Regeln gelten auch für die späteren Lektionen:

- Wenn das Buch PowerShell im Projektordner verlangt, öffnest du auf dem Mac das Terminal in diesem Ordner. Der aktuelle Ordner muss auf beiden Systemen stimmen.
- In Terminalbefehlen wird aus `.\uebung.js` die Schreibweise `./uebung.js`. Ebenso wird aus `.\src\app.js` die Schreibweise `./src/app.js`. In JavaScript-Strings und in import-Anweisungen veränderst du nichts nach dieser Regel.
- Windows verwendet `npm.cmd`, der Mac `npm`. Aus `npm.cmd init -y` wird `npm init -y`, aus `npm.cmd pkg set type=module` wird `npm pkg set type=module`.
- Speichern und Kopieren erfolgen auf dem Mac gewöhnlich mit Befehl statt Strg. Zum Abbrechen eines laufenden Terminalprogramms bleibt es auf beiden Systemen bei Strg+C.
- Für die Konsole einer Browserseite kannst du die Entwicklerwerkzeuge über das Browsermenü öffnen. In Safari musst du die Funktionen für Webentwickler gegebenenfalls zuerst in den Einstellungen aktivieren.

Die KI soll bei Einrichtungsfragen dein Betriebssystem kennen. Wenn ein Windows-Befehl auf deinem Mac nicht funktioniert, gib die Buchstelle und deinen aktuellen Ordner an und lass dir die passende Variante erklären.

Dein Lernsystem

So beginnt jede Lerneinheit

Öffne deinen Lernordner in VS Code; unter Windows kannst du dafür das eingerichtete Desktop-Symbol benutzen. Klicke auf `uebung.js`. Für eine neue, unabhängige Aufgabe markierst du dort den alten Inhalt und löschst ihn. Bei einem Projekt aus mehreren Lektionen bleibt der benötigte Projektcode erhalten.

Tippe den neuen Code in die Datei, speichere und starte ihn. Die Ausgabe steht unten im Bereich Ausgabe von Code Runner. Vergleiche sie mit deiner Erwartung. Wenn du etwas änderst, führe die Datei erneut aus; die Anzeige aktualisiert sich nicht schon beim Tippen.

1. Lies die Lektion und erkläre dir den Ablauf.
2. Tippe das Beispiel selbst und führe es aus.
3. Ändere einen Wert und sage die neue Ausgabe voraus.
4. Versuche die Buchübung ohne Musterlösung.
5. Bearbeite zusätzliche KI-Aufgaben mit dem Lektionsprompt.
6. Wiederhole am nächsten Lerntag einen älteren Inhalt aus dem Gedächtnis.

Wenn ein Fehler auftaucht

Lies die Meldung und prüfe zuerst die genannte Datei und Zeile. Suche nach einem fehlenden Zeichen, einem falsch geschriebenen Namen oder einer unpassenden Bedingung. Ändere möglichst nur eine Sache und starte erneut. Wenn du nicht weiterkommst, bitte die KI um einen kleinen Hinweis. Eine vollständige Lösung lässt du dir erst zeigen, wenn du sie ausdrücklich brauchst.

Deinen Lernstand festhalten

Notiere nach einer Lektion kurz, was du selbst geschrieben hast und wo du Hilfe gebraucht hast. Eine richtige Lösung nach Vorgabe und eine selbst entwickelte Lösung sind unterschiedliche Lernstände. Die Zwischenstände im Kurs helfen dir, ältere Themen zu kombinieren und offene Punkte zu erkennen.

Wenn die KI eine unbekannte Methode benutzt, lass sie die Aufgabe auf deine bereits gelernten Befehle beschränken. Gib bei Bedarf die Begriffstabelle der Lektion dazu. Nimm dir Zeit für Wiederholung, bevor du ein neues Werkzeug ergäntzt.

Die Inhaltsübersicht dieser Leseprobe steht auf Seite 3.

Es folgen die Lektionen 1 bis 24. Anschließend kannst du mit einem eigenen Abschlusscheck mehrere Grundlagen verbinden.

Teil A Die ersten Schritte

Hier entsteht das Fundament. Neue Zeichen werden bewusst langsam eingeführt, damit du Code nicht nur kopierst, sondern lesen kannst.

Lektion 1 Was ist Programmcode

Worum geht es

Programmcode ist eine Folge eindeutiger Anweisungen. Der Computer arbeitet nach Regeln und ergänzt keine Absicht, die nicht im Code steht.

Warum brauchst du das

Wenn du Code als kleine Arbeitsschritte liest, wirkt er weniger wie eine Textwand. Diese Lesefähigkeit brauchst du später beim Schreiben, Prüfen und Debuggen.

Ganz langsam gedacht

Eine JavaScript-Datei ist zunächst einfach eine Textdatei mit Anweisungen. Du schreibst sie. Erst wenn Node.js oder ein Browser sie ausführt, passiert etwas. Nur weil Code in VS Code steht, läuft er noch nicht.

JavaScript unterscheidet Groß- und Kleinschreibung. `console` und `Console` wären zwei verschiedene Namen. Später wirst du merken: Schon ein einzelner falscher Buchstabe kann eine Fehlermeldung auslösen.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
Anweisung	Ein einzelner Arbeitsauftrag im Programm.
Programm	Eine geordnete Menge von Anweisungen.
//	Beginnt einen Kommentar bis zum Zeilenende.
;	Kennzeichnet häufig das Ende einer Anweisung.

Beispiel

```
// Zwei Anweisungen  
console.log("Start");  
console.log("Fertig");
```

Was passiert Schritt für Schritt

- Die erste Zeile ist ein Kommentar und wird nicht ausgeführt.
- JavaScript führt die erste `console.log`-Anweisung aus.
- Danach folgt die zweite Anweisung.

- Die Reihenfolge im Quelltext bestimmt hier die Reihenfolge der Ausgabe.

Deine Übung

Schreibe drei `console.log`-Zeilen, die nacheinander "Eins", "Zwei" und "Drei" ausgeben. Füge darüber einen Kommentar ein, der beschreibt, was das Programm tut.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 1: "Was ist Programmcode?". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: Kommentare, `console.log` und einfache Textausgaben. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: Lies Code zunächst Zeile für Zeile: Was bekommt JavaScript hier konkret als Auftrag?

Musterlösung 1: erst nach deinem Versuch prüfen

```
// Drei Ausgaben
console.log("Eins");
console.log("Zwei");
console.log("Drei");
```

Lektion 2 Das erste JavaScript-Programm

Worum geht es

Du führst eine JavaScript-Datei selbst aus und zerlegst eine einfache Ausgabe in ihre einzelnen Zeichen und Bestandteile.

Warum brauchst du das

Wer versteht, was bei `console.log("Hallo Welt");` jedes Zeichen bedeutet, kann später neue Befehle viel leichter lesen.

Jedes Zeichen hat eine Aufgabe

In `console.log("Hallo Welt");` ist nichts Dekoration: `console` ist der Name eines eingebauten Werkzeugs, der Punkt greift auf etwas darin zu, `log` ist die Ausgabe-Funktion, die runden Klammern enthalten die Übergabe, die Anführungszeichen markieren Text und das Semikolon beendet die Anweisung.

Du musst das Semikolon in modernem JavaScript nicht an jeder Stelle zwingend schreiben. In diesem Buch verwenden wir es trotzdem konsequent, weil es für Anfänger die Grenzen von Anweisungen sichtbar macht.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
console	Ein eingebautes Objekt für Konsolenausgaben.
.	Punktnotation: wählt etwas an einem Objekt aus.
log	Eine Methode des console-Objekts.
(...)	Runde Klammern enthalten die übergebenen Werte.
"..."	Anführungszeichen kennzeichnen Text.
;	Ende der Anweisung.

Beispiel

```
console.log("Hallo Welt");
```

Was passiert Schritt für Schritt

- console bezeichnet das Konsolen-Werkzeug.
- Der Punkt führt zu der Methode log.
- Die runden Klammern enthalten den Wert, der ausgegeben werden soll.
- "Hallo Welt" ist ein String, also Text.
- Beim Ausführen erscheint Hallo Welt in der Konsole.

Deine Übung

Lösche den alten Inhalt von uebung.js. Gib anschließend deinen Vornamen und darunter den Text "Ich lerne JavaScript" aus. Schreibe beide Anweisungen selbst.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 2: "Das erste JavaScript-Programm". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: console.log, Strings und Kommentare. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: console.log zeigt einen Wert an. Die Klammern enthalten, was ausgegeben werden soll.

Musterlösung 2: erst nach deinem Versuch prüfen

```
console.log("Anna");  
console.log("Ich lerne JavaScript");
```

Lektion 3 Werte und Datentypen

Worum geht es

JavaScript verarbeitet unterschiedliche Arten von Werten. Zunächst reichen drei besonders wichtige Typen: Zahl, Text und Wahrheitswert.

Warum brauchst du das

Der Datentyp beeinflusst, was JavaScript mit einem Wert tun kann. Die Zahl 10 und der Text "10" sehen ähnlich aus, sind aber nicht dasselbe.

Zahl 2 und Text "2" sind nicht dasselbe

Die Zahl 2 kannst du direkt zum Rechnen verwenden. Ein String ist eine Zeichenkette, also eine Folge von Zeichen. Wir sagen dazu anfangs auch Text. Ein String kann jedoch genauso Ziffern enthalten: "2" besteht aus einer Ziffer, ist durch die Anführungszeichen aber ein String und keine Zahl.

```
console.log(2 + 2);      // 4  
console.log("2" + "2"); // 22
```

Bei zwei Zahlen addiert +. Ist bei unseren einfachen Werten eine Seite ein String, verbindet + beide als Text. Deshalb ergibt "2" + 3 den String "23". Eine Benutzereingabe ist nicht automatisch eine Rechenzahl.

JavaScript wandelt Werte in manchen Situationen automatisch um. Dafür begegnet dir der Begriff Typumwandlung oder Coercion. Für diese ersten Übungen verwenden wir beim Rechnen bewusst Zahlen. Wie du einen Text selbst mit Number in eine Zahl umwandelst, lernst du in Lektion 45.

Kommazahlen verwenden einen Punkt

Im JavaScript-Code schreibst du 3.5 und nicht 3,5. Das Komma hat in JavaScript andere Aufgaben, zum Beispiel mehrere Werte voneinander zu trennen.

```
console.log(3.5 + 1.5); // 5
```

Später lernst du, wie du deutsche Eingaben wie "3,50" in eine JavaScript-Zahl umwandelst. Für den Anfang gilt: Zahlen im Code immer mit Punkt.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
number	Zahl, zum Beispiel 42 oder 3.5.
string	Text beziehungsweise Zeichenkette.

Begriff / Zeichen	Bedeutung
boolean	Wahrheitswert mit genau zwei Möglichkeiten.
true / false	Die beiden Boolean-Werte.

Beispiel

```
console.log(42);  
console.log("42");  
console.log(true);
```

Was passiert Schritt für Schritt

- 42 ist ein number.
- "42" steht in Anführungszeichen und ist deshalb ein string.
- true steht ohne Anführungszeichen und ist ein boolean.
- Alle drei Werte können mit console.log angezeigt werden, behalten aber ihren jeweiligen Typ.

Deine Übung

Gib nacheinander dein Alter als Zahl, deinen Namen als String und einen Boolean aus, der ausdrückt, ob du heute übst.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 3: "Werte und Datentypen". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: console.log sowie number, string und boolean. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: Ein Wert hat nicht nur Inhalt, sondern auch einen Datentyp.

Musterlösung 3: erst nach deinem Versuch prüfen

```
console.log(30);  
console.log("Anna");  
console.log(true);
```

Lektion 4 Variablen mit const

Worum geht es

Mit einer Variablen gibst du einem Wert einen Namen. const verwendest du, wenn diese Variablenzuweisung später nicht ersetzt werden soll.

Warum brauchst du das

Benannte Werte machen Code lesbarer und verhindern, dass dieselbe Information an vielen Stellen erneut geschrieben werden muss.

Eine Variable ist ein Name, keine Schachtel aus Magie

Du kannst dir `const preis = 30;` zunächst wie ein beschriftetes Schild vorstellen: Der Name `preis` verweist auf den Wert 30. Später werden Variablen auch auf Arrays und Objekte verweisen. Deshalb ist die Vorstellung einer einfachen "Schachtel" nützlich, aber nicht für jeden späteren Fall technisch exakt.

Variablennamen dürfen keine Leerzeichen enthalten. Verwende für den Anfang einfache Namen wie `preis`, `anzahl` oder `gesamtPreis`. JavaScript unterscheidet auch hier Groß- und Kleinschreibung: `preis` und `Preis` wären zwei verschiedene Namen.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
<code>const</code>	Legt eine Variable an, die später nicht neu zugewiesen werden kann.
Variable	Ein Name, unter dem ein Wert erreichbar ist.
<code>=</code>	Zuweisungsoperator: rechts wird ausgewertet und links gespeichert.
Bezeichner	Der Name einer Variable, Funktion oder ähnlichen Struktur.

Beispiel

```
const preis = 30;  
console.log(preis);
```

Was passiert Schritt für Schritt

- JavaScript legt den Namen `preis` an.
- Der Operator `=` weist diesem Namen den Wert 30 zu.
- `console.log(preis)` liest den gespeicherten Wert.
- Die Ausgabe ist 30.

Wähle Namen, die mit einem Buchstaben beginnen, und verwende danach Buchstaben und Ziffern. Leerzeichen und Zeichen wie `&` sind hier nicht erlaubt. Schlüsselwörter wie `const` sind reserviert. `gesamtPreis` ist ein Beispiel für camelCase: weitere Wörter beginnen groß. Die Namen bestimmst du selbst.

Deine Übung

Lege drei const-Variablen für einen Produktnamen, einen Preis und die Information an, ob das Produkt verfügbar ist. Gib alle drei Werte aus.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 4: "Variablen mit const". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: const, einfache Werte und console.log. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: Eine Variable gibt einem Wert einen verständlichen Namen; const schützt die Zuweisung vor versehentlichem Ersetzen.

Musterlösung 4: erst nach deinem Versuch prüfen

```
const produktName = "Tasse";
const preis = 12;
const verfuegbar = true;
console.log(produktName);
console.log(preis);
console.log(verfuegbar);
```

Lektion 5 Veränderliche Werte mit let

Worum geht es

let legt eine Variable an, deren Wert später durch eine neue Zuweisung ersetzt werden darf.

Warum brauchst du das

Viele Zustände verändern sich während eines Programms: Guthaben sinkt, Punkte steigen oder ein Zähler wird erhöht. Dafür brauchst du veränderliche Variablen.

Warum nicht einfach immer let

Weil const dem Leser und dir selbst zeigt: Diese Zuweisung soll nicht versehentlich ersetzt werden. Wenn ein Wert wirklich wechseln muss, nimm let. Diese kleine Gewohnheit macht später größeren Code leichter verständlich.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
let	Legt eine später neu zuweisbare Variable an.
Neuzuweisung	Eine vorhandene Variable erhält einen neuen Wert.

Begriff / Zeichen	Bedeutung
=	Setzt einen Wert; es bedeutet nicht mathematisch "ist gleich".

Beispiel

```
let geld = 100;  
geld = 70;  
console.log(geld);
```

Was passiert Schritt für Schritt

- geld wird mit 100 angelegt.
- Die zweite Zeile legt keine neue Variable an; sie ersetzt den gespeicherten Wert durch 70.
- console.log zeigt deshalb 70.

Deine Übung

Lege `let punkte = 10` an. Weise danach 15 zu und gib den neuen Wert aus. Lege zusätzlich eine `const`-Variable `spielName` an.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 5: "Veränderliche Werte mit let". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: `const`, `let`, Zuweisung und `console.log`. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: `const` bleibt auf derselben Zuweisung; `let` darf später einen neuen Wert bekommen.

Musterlösung 5: erst nach deinem Versuch prüfen

```
let punkte = 10;  
punkte = 15;  
const spielName = "Abenteuer";  
console.log(punkte);
```

Lektion 6 Rechnen mit JavaScript

Worum geht es

JavaScript kann Zahlen addieren, subtrahieren, multiplizieren, dividieren und Reste einer Division bestimmen.

Warum brauchst du das

Berechnungen sind in fast jedem Programm enthalten - von Preisen und Punkteständen bis zu Messwerten oder Statistiken.

Achtung bei Text und Rechnen

```
console.log(2 + 3);      // 5
console.log("2" + 3);    // "23"
// Bewusste Textumwandlung folgt in Lektion 45.
```

Andere Rechenoperatoren versuchen mitunter, Text umzuwandeln: "2" * 3 ergibt 6. Verwende hier ausschließlich Zahlen. Die bewusste Umwandlung mit Number lernst du in Lektion 45.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
+	Addition.
-	Subtraktion.
*	Multiplikation.
/	Division.
%	Rest einer Division; bei negativen Zahlen nicht gleich mathematischem Modulo.

Beispiel

```
const preis = 12;
const anzahl = 3;
const gesamt = preis * anzahl;
console.log(gesamt);
```

Was passiert Schritt für Schritt

- preis enthält 12.
- anzahl enthält 3.
- preis * anzahl ergibt 36.
- Das Ergebnis wird als gesamt gespeichert und danach ausgegeben.

Deine Übung

Ein Ticket kostet 18 Euro. Lege Preis und Anzahl für vier Tickets als Variablen an, berechne den Gesamtpreis und gib ihn aus. Berechne zusätzlich mit dem Operator % den Rest der Division 17 durch 5.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 6: "Rechnen mit JavaScript". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: Variablen, Zahlen und + - * / %. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: Rechenausdrücke können wie andere Werte gespeichert und weiterverwendet werden.

Musterlösung 6: erst nach deinem Versuch prüfen

```
const preis = 18;
const anzahl = 4;
console.log(preis * anzahl); // 72
console.log(17 % 5); // 2
```

Lektion 7 Reihenfolge und Klammern

Worum geht es

Rechenausdrücke werden nach festen Vorrangregeln ausgewertet. Runde Klammern können die Reihenfolge ausdrücklich festlegen.

Warum brauchst du das

Ein korrekt geschriebener Ausdruck kann trotzdem ein falsches Ergebnis liefern, wenn du die Reihenfolge der Berechnung falsch einschätzt.

Klammern sind eine Anweisung an die Reihenfolge

JavaScript rechnet wie in der Schule Multiplikation und Division vor Addition und Subtraktion. Runde Klammern können diese Reihenfolge ändern. Wenn du unsicher bist, schreibe Teilergebnisse in eigene Variablen. Das ist oft lesbarer als ein besonders kurzer Ausdruck.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
Vorrang	Regel, welcher Teil eines Ausdrucks zuerst berechnet wird.
(...)	Erzwingt beziehungsweise verdeutlicht eine gewünschte Reihenfolge.

Beispiel

```
const preis = 20;  
const anzahl = 3;  
const versand = 5;  
const gesamt = (preis * anzahl) + versand;  
console.log(gesamt);
```

Was passiert Schritt für Schritt

- Zuerst wird `preis * anzahl` berechnet: 60.
- Die Klammern machen diese Gruppe besonders deutlich.
- Danach werden 5 Versandkosten addiert.
- `gesamt` enthält 65.

Deine Übung

Berechne den Preis für 5 Artikel zu je 8 Euro plus 6 Euro Versand. Berechne danach bewusst einen Ausdruck mit Klammern, der 10 minus 2 zuerst berechnet und das Ergebnis mit 3 multipliziert.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 7: "Reihenfolge und Klammern". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: Grundrechenarten, Variablen und Klammern. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: Wenn ein Ausdruck nicht sofort eindeutig lesbar ist, helfen Klammern sowohl JavaScript als auch dem Menschen.

Musterlösung 7: erst nach deinem Versuch prüfen

```
console.log(5 * 8 + 6); // 46  
console.log((10 - 2) * 3); // 24
```

Lektion 8 Vergleiche

Worum geht es

Vergleichsoperatoren stellen Fragen an Werte. Das Ergebnis eines Vergleichs ist immer `true` oder `false`.

Warum brauchst du das

Bedingungen in Programmen bauen auf solchen Wahrheitswerten auf. Bevor `if` sinnvoll ist, musst du verstehen, wie diese Fragen entstehen.

Vergleiche liefern immer einen Wahrheitswert

Ein Vergleich ist keine Ausgabe und keine Entscheidung. Er erzeugt zunächst nur `true` oder `false`. Erst später benutzt `if` diesen Wahrheitswert, um einen Codeweg auszuwählen.

Wir verwenden `===` statt `==`. Das dreifache Gleichheitszeichen prüft Wert und Typ. Dadurch vermeidest du automatische Umwandlungen, die bei `==` zu überraschenden Ergebnissen führen können.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
<code>===</code>	Streng gleich: Wert und Typ müssen übereinstimmen.
<code>!==</code>	Streng ungleich.
<code>></code> / <code><</code>	Größer als / kleiner als.
<code>>=</code> / <code><=</code>	Größer oder gleich / kleiner oder gleich.

Beispiel

```
const geld = 50;
console.log(geld >= 50);
console.log(geld === 50);
console.log(geld < 20);
```

Was passiert Schritt für Schritt

- `50 >= 50` ist `true`, weil der Grenzwert eingeschlossen ist.
- `50 === 50` ist ebenfalls `true`.
- `50 < 20` ist `false`.
- Jeder Vergleich liefert einen Boolean.

Deine Übung

Lege eine Variable `alter` an. Gib aus, ob das Alter mindestens 18 ist, genau 18 ist und kleiner als 18 ist. Ändere anschließend den Wert und beobachte die Ausgaben.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 8: "Vergleiche". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: Variablen, Datentypen, Rechnen und Vergleichsoperatoren. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: Vergleiche verändern keinen Wert; sie beantworten eine Frage mit `true` oder `false`.

Musterlösung 8: erst nach deinem Versuch prüfen

```
let alter = 18;
console.log(alter >= 18);
console.log(alter === 18);
console.log(alter < 18);
alter = 17;
console.log(alter >= 18); // false
```

Teil B Entscheidungen und Logik

Programme werden nützlich, wenn sie abhängig von Daten unterschiedliche Wege nehmen können.

Lektion 9 if Code nur bei einer Bedingung ausführen

Worum geht es

Mit if wird ein Codeblock nur ausgeführt, wenn eine Bedingung true ergibt.

Warum brauchst du das

So reagiert ein Programm auf Daten, statt immer starr dieselben Anweisungen auszuführen.

Lies if (temperatur >= 20) zunächst laut als deutschen Satz: "Wenn Temperatur größer oder gleich 20 ist ...". Diese Übersetzung hilft vielen Anfängern mehr als Auswendiglernen.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
if	Bedeutet sinngemäß "wenn".
(Bedingung)	Prüfung, deren Wert als wahr oder falsch ausgewertet wird.
{ ... }	Der Codeblock, der bei true ausgeführt wird.

Beispiel

```
const geld = 80;  
  
if (geld >= 50) {  
    console.log("Kauf möglich");  
}
```

Was passiert Schritt für Schritt

- JavaScript prüft `geld >= 50`.
- `80 >= 50` ergibt `true`.
- Darum wird der Block zwischen den geschweiften Klammern ausgeführt.
- Wäre die Bedingung `false`, würde der Block übersprungen.

In diesem Abschnitt verwenden wir Booleans und Vergleiche. Allgemein wertet if auch andere Werte aus: 0 und ein leerer String gelten beispielsweise als falsch. Deshalb

sagen wir hier vereinfacht „bei true“. Schreibe die gewünschte Frage anfangs ausdrücklich als Vergleich.

Deine Übung

Lege eine Variable temperatur an. Gib "Warm genug" nur dann aus, wenn die Temperatur mindestens 20 beträgt. Teste anschließend einen Wert darunter.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 9: "if - Code nur bei einer Bedingung ausführen". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: Vergleiche, Boolean, Variablen und if. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: if führt seinen Block nur bei true aus.

Musterlösung 9: erst nach deinem Versuch prüfen

```
const temperatur = 20;
if (temperatur >= 20) {
  console.log("Warm genug");
}
```

Lektion 10 else der alternative Weg

Worum geht es

else ergänzt eine if-Abfrage um einen zweiten Weg, der bei false ausgeführt wird.

Warum brauchst du das

Viele Entscheidungen haben zwei sinnvolle Ergebnisse: genug oder zu wenig, erlaubt oder nicht erlaubt, vorhanden oder nicht vorhanden.

Bei if/else läuft immer genau einer der beiden Blöcke. Nicht beide, und auch nicht keiner.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
else	Bedeutet sinngemäß "sonst".
if ... else	Genau einer der beiden Blöcke wird ausgeführt.

Beispiel

```
const geld = 35;

if (geld >= 50) {
  console.log("Genug Geld");
} else {
  console.log("Zu wenig Geld");
}
```

Was passiert Schritt für Schritt

- `geld >= 50` ergibt `false`.
- Der `if`-Block wird übersprungen.
- JavaScript wechselt in den `else`-Block.
- Die Ausgabe lautet "Zu wenig Geld".

Deine Übung

Prüfe mit `if/else`, ob ein Punktestand mindestens 100 beträgt. Gib passend "Level geschafft" oder "Noch nicht geschafft" aus.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 10: "else - der alternative Weg". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: `if`, `else`, Vergleiche und `console.log`. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: `if` behandelt `true`; `else` behandelt den übrigen Fall.

Musterlösung 10: erst nach deinem Versuch prüfen

```
const punkte = 100;
if (punkte >= 100) {
  console.log("Level geschafft");
} else {
  console.log("Noch nicht geschafft");
}
```

Lektion 11 else if mehrere Fälle unterscheiden

Worum geht es

Mit `else if` kannst du mehrere Bedingungen der Reihe nach prüfen.

Warum brauchst du das

Reale Regeln bestehen oft aus mehr als zwei Bereichen, zum Beispiel Bewertung nach Punktestufen oder Versandkosten nach Gewicht.

else if wird von oben nach unten geprüft. Sobald eine Bedingung true ist, werden die späteren Zweige dieser Kette übersprungen.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
else if	Weitere Bedingung, wenn vorherige Bedingungen false waren.
Reihenfolge	Prüfungen laufen von oben nach unten; der erste passende Zweig gewinnt.

Beispiel

```
const punkte = 82;

if (punkte >= 90) {
  console.log("Sehr gut");
} else if (punkte >= 70) {
  console.log("Gut");
} else {
  console.log("Weiter üben");
}
```

Was passiert Schritt für Schritt

- `82 >= 90` ist false.
- Danach wird `82 >= 70` geprüft und ergibt true.
- Dieser Block läuft.
- Spätere Zweige werden nicht mehr geprüft.

Deine Übung

Erstelle drei Altersbereiche: unter 13, 13 bis 17 und ab 18. Gib für jeden Bereich einen passenden Text aus und teste mindestens drei Werte.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 11: "else if - mehrere Fälle unterscheiden". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: if, else, else if und Vergleiche. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: Bei einer else-if-Kette zählt die Reihenfolge: Der erste passende Zweig wird ausgeführt.

Musterlösung 11: erst nach deinem Versuch prüfen

```
const alter = 15;
if (alter < 13) {
  console.log("Unter 13");
} else if (alter < 18) {
  console.log("13 bis 17");
} else {
  console.log("Ab 18");
}
```

Lektion 12 && Bedingungen mit UND verbinden

Worum geht es

Der Operator `&&` verlangt, dass mehrere Bedingungen gleichzeitig true sind.

Warum brauchst du das

Damit kannst du Regeln formulieren, bei denen mehrere Voraussetzungen gemeinsam erfüllt sein müssen.

`&&` bedeutet nicht "und dann", sondern logisch: Beide Seiten müssen true sein.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
<code>&&</code>	Logisches UND. Beide Seiten müssen true sein.
Teilbedingung	Eine einzelne Frage innerhalb einer größeren Bedingung.

Beispiel

```
const geld = 80;
const hatTicket = true;

if (geld >= 50 && hatTicket === true) {
  console.log("Zugang möglich");
}
```

Was passiert Schritt für Schritt

- `geld >= 50` ergibt true.
- `hatTicket === true` ergibt ebenfalls true.
- `true && true` ergibt true.

- Der if-Block wird ausgeführt.

Hier verbinden wir ausschließlich Boolean-Werte. Allgemein liefert `&&` einen der ursprünglichen Werte und nicht immer einen Boolean. Außerdem wird rechts nur geprüft, wenn links als wahr gilt. Das nennt sich Kurzschlussauswertung.

Deine Übung

Prüfe, ob eine Person mindestens 18 Jahre alt ist UND einen Ausweis dabei hat. Lege beide Werte als Variablen an und teste auch einen Fall, in dem nur eine Bedingung stimmt.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 12: "&& - Bedingungen mit UND verbinden". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: if/else, Vergleiche, Boolean und &&. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: `&&` bedeutet: Alle verbundenen Bedingungen müssen stimmen.

Musterlösung 12: erst nach deinem Versuch prüfen

```
const alter = 18;
const hatAusweis = true;
if (alter >= 18 && hatAusweis) {
  console.log("Erlaubt");
}
```

Lektion 13 || Bedingungen mit ODER verbinden

Worum geht es

Der Operator `||` ergibt true, wenn mindestens eine der verbundenen Bedingungen true ist.

Warum brauchst du das

ODER-Regeln brauchst du, wenn mehrere unterschiedliche Wege zum gleichen Ergebnis führen können.

`||` bedeutet: Mindestens eine Seite muss true sein. Sind beide true, ist das Gesamtergebnis ebenfalls true.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
<code> </code>	Logisches ODER. Mindestens eine Seite muss true sein.

Beispiel

```
const istMitglied = false;
const hatGutschein = true;

if (istMitglied || hatGutschein) {
  console.log("Rabatt möglich");
}
```

Was passiert Schritt für Schritt

- istMitglied ist false.
- hatGutschein ist true.
- false || true ergibt true.
- Der Rabatt-Zweig wird ausgeführt.

Für unsere Booleans gilt diese ODER-Regel. Allgemein liefert || einen der ursprünglichen Werte. Ist links bereits wahr, wird rechts nicht ausgewertet. && hat höheren Vorrang als ||; klammere gemischte Regeln ausdrücklich.

Deine Übung

Lege zwei Booleans an: hatSchluessel und tuerIstOffen. Gib "Du kannst hinein" aus, wenn mindestens eine der beiden Bedingungen true ist.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 13: "|| - Bedingungen mit ODER verbinden". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: if/else, Boolean, Vergleiche, && und ||. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: || bedeutet: Mindestens eine Möglichkeit reicht aus.

Musterlösung 13: erst nach deinem Versuch prüfen

```
const hatSchluessel = false;
const tuerIstOffen = true;
if (hatSchluessel || tuerIstOffen) {
  console.log("Du kannst hinein");
}
```

Lektion 14 ! einen Wahrheitswert umkehren

Worum geht es

Das Ausrufezeichen vor einem Wert kehrt dessen Wahrheitsbedeutung um: true wird false und false wird true.

Warum brauchst du das

Damit lassen sich Bedingungen oft natürlicher lesen, etwa "wenn nicht gesperrt".

! steht direkt vor einem Wert oder Ausdruck und kehrt dessen Wahrheitswert um. ! false wird true; !true wird false.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
!	Logisches NICHT.
!wert	Kehrt die Wahrheitsbedeutung des folgenden Wertes um.
!==	Ein anderer Operator: bedeutet "streng ungleich".

Beispiel

```
const gesperrt = false;

if (!gesperrt) {
  console.log("Zugang erlaubt");
}
```

Was passiert Schritt für Schritt

- gesperrt enthält false.
- !gesperrt macht daraus true.
- Die if-Bedingung ist damit erfüllt.
- Das ist nicht dasselbe wie der Vergleichsoperator !==.

Deine Übung

Lege online = false an. Gib mit ! eine Meldung aus, wenn die Person nicht online ist. Ändere den Wert danach auf true und teste erneut.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 14: "! - einen Wahrheitswert umkehren". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: Boolean, if/else, &&, || und !. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach

gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: `!` dreht einen Wahrheitswert um; `!==` vergleicht zwei Werte auf Ungleichheit.

Musterlösung 14: erst nach deinem Versuch prüfen

```
let online = false;
if (!online) {
  console.log("Nicht online");
}
online = true;
if (!online) {
  console.log("Nicht online");
}
```

Lektion 15 Verschachtelte Bedingungen

Worum geht es

Ein `if`-Block kann selbst wieder eine weitere Bedingung enthalten. Das nennt man Verschachtelung.

Warum brauchst du das

Manchmal hängt eine zweite Entscheidung sinnvoll von einer ersten ab. Gleichzeitig lernst du hier, warum tiefe Verschachtelung schnell unübersichtlich wird.

Verschachtelte `if`-Blöcke sind erlaubt, aber sie werden schnell schwer lesbar. Wenn du drei oder vier Ebenen ineinander baust, ist das oft ein Zeichen, dass du später eine bessere Struktur brauchst.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
Verschachtelung	Ein Codeblock liegt innerhalb eines anderen Blocks.
Einrückung	Optische Struktur, die zusammengehörigen Code sichtbar macht.

Beispiel

```
const eingeloggt = true;
const admin = false;

if (eingeloggt) {
  if (admin) {
    console.log("Adminbereich");
  } else {
    console.log("Normaler Bereich");
  }
}
```

Was passiert Schritt für Schritt

- Zuerst wird geprüft, ob eingeloggt true ist.
- Nur dann erreicht JavaScript die innere Bedingung.
- Dort wird admin geprüft.
- Weil admin false ist, läuft der innere else-Zweig.

Deine Übung

Prüfe zuerst, ob ein Spieler noch Lebenspunkte hat. Nur wenn das stimmt, prüfst du innen, ob genug Energie für eine Spezialaktion vorhanden ist.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 15: "Verschachtelte Bedingungen". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: if/else, logische Operatoren und verschachtelte Blöcke. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: Verschachtelung ist erlaubt, aber jede zusätzliche Ebene sollte einen klaren Grund haben.

Musterlösung 15: erst nach deinem Versuch prüfen

```
const leben = 50;
const energie = 20;
if (leben > 0) {
  if (energie >= 10) {
    console.log("Spezialaktion möglich");
  } else {
    console.log("Zu wenig Energie");
  }
}
```

Lektion 16 Bedingungen sinnvoll planen

Worum geht es

Bevor du komplizierte Bedingungen tippst, formulierst du die Regel in normaler Sprache und zerlegst sie in einzelne Fragen.

Warum brauchst du das

Viele Logikfehler entstehen nicht an der Syntax, sondern weil die Regel selbst unklar formuliert wurde.

Erst die Regel, dann der Code

Schreibe bei schwierigeren Bedingungen zuerst einen deutschen Satz oder kleine Teilregeln. Übersetze erst danach in `&&`, `||` und `!`. Programmieren besteht sehr oft genau aus diesem Übersetzen von Anforderungen in eindeutige Logik.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
Anforderung	Was das Programm fachlich tun soll.
Teilbedingung	Eine einzelne prüfbare Aussage.
Grenzwert	Ein Wert, an dem sich das Verhalten ändert.

Beispiel

```
const alter = 20;
const ticket = true;
const gesperrt = false;

if (alter >= 18 && ticket && !gesperrt) {
  console.log("Einlass");
} else {
  console.log("Kein Einlass");
}
```

Was passiert Schritt für Schritt

- Regel in Alltagssprache: mindestens 18 UND Ticket UND nicht gesperrt.
- Jeder Teil bekommt eine eigene Variable oder einen Vergleich.
- Die Teilbedingungen werden mit `&&` verbunden.
- Danach testest du bewusst Grenz- und Gegenfälle.

Deine Übung

Formuliere zuerst in einem Kommentar eine Regel für kostenlosen Versand: mindestens 50 Euro Warenwert ODER Premiumkunde, aber der Einkauf darf nicht gesperrt sein. Setze die Regel danach in Code um.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 16: "Bedingungen sinnvoll planen". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: Variablen, Datentypen, Rechnen, Vergleiche, if/else, &&, ||, ! und Klammern. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: Gute Bedingungen beginnen mit einer klaren Regel in Alltagssprache.

Musterlösung 16: erst nach deinem Versuch prüfen

```
// Ab 50 oder Premium, aber nicht gesperrt.
const warenwert = 40;
const premium = true;
const gesperrt = false;
if ((warenwert >= 50 || premium) && !gesperrt) {
  console.log("Kostenloser Versand");
} else {
  console.log("Versand kostenpflichtig");
}
```

Zwischenstand 1 Grundlogik sitzt sie wirklich

Dieser Abschnitt zählt nicht als neue Lektion. Er soll verhindern, dass du nur weiterblätterst, obwohl ältere Grundlagen noch wackeln.

Mischaufgabe 1: Ein Warenkorb kostet 48 Euro, Versand 5 Euro. Kostenloser Versand gilt ab 50 Euro oder für Premiumkunden, aber nie für gesperrte Einkäufe. Berechne den Endpreis. Teste 49, 50, Premium und gesperrt.

Mischaufgabe 2: Schreibe die Einlassregel: ab 18, Ticket vorhanden, nicht gesperrt. Teste 17 und 18 und erkläre jede Teilbedingung.

Notiere für jede Aufgabe: ohne Hilfe geschrieben, nachgeschlagen oder mit Hinweisen gelöst. Erkläre anschließend den Ablauf und ändere einen Testwert. Eine richtige Lösung nach Vorgabe belegt noch keine selbstständige Umsetzung.

KI-Mischtraining und Lernstandsdiagnose

Ich habe in diesem JavaScript-Buch bisher folgende Themen gelernt: Variablen, Rechnen, Vergleiche, if/else und logische Operatoren. Prüfe meinen Lernstand mit gemischten Aufgaben. Stelle immer nur eine Aufgabe gleichzeitig und gib keine Lösung vor. Beginne leicht, mische dann mehrere bereits gelernte Themen. Wenn ich einen Fehler mache, gib zunächst nur einen kleinen Hinweis. Nach 6 Aufgaben fasse meinen Stand in drei Bereichen zusammen: "sitzt", "unsicher", "noch üben". Begründe jeden Punkt kurz anhand meiner Antworten. Gib mir danach 3 gezielte Wiederholungsaufgaben nur zu den unsicheren Themen. Nutze keine Inhalte, die im genannten Lernstoff noch nicht vorkamen. Trenne in der Diagnose ausdrücklich zwischen erklärt/versteht und ohne Lösungsvorgabe selbst geschrieben. Bewerte nur beobachtete Leistungen und nenne nicht geprüfte Themen als offen.

Wichtig: Die Einschätzung der KI ist nur eine Lernhilfe. Entscheidend ist, ob du den Code selbst schreiben, erklären und verändern kannst.

Teil C Arrays und Schleifen

Jetzt arbeitest du nicht mehr nur mit einzelnen Werten, sondern mit ganzen Sammlungen.

Lektion 17 Arrays mehrere Werte unter einem Namen

Worum geht es

Ein Array speichert mehrere Werte in einer geordneten Liste.

Warum brauchst du das

Listen brauchst du überall dort, wo nicht nur ein einzelner Wert, sondern viele ähnliche Werte verarbeitet werden sollen.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
Array	Geordnete Liste von Werten.
[...]	Eckige Klammern erzeugen ein Array.
,	Trennt die einzelnen Elemente.

Beispiel

```
const tiere = ["Hund", "Katze", "Maus"];  
console.log(tiere);
```

Was passiert Schritt für Schritt

- Die eckigen Klammern kennzeichnen das Array.
- Drei Strings werden durch Kommas getrennt.
- Die Variable `tiere` verweist auf die gesamte Liste.
- `console.log` kann das komplette Array anzeigen.

Arrays sind Objekte. `const` verhindert, dass `tiere` später eine andere Liste zugewiesen wird; vorhandene Einträge dürfen sich ändern. Ersetzen mit `tiere[0] = "Fuchs"` ist eine Inhaltsänderung. Mit `tiere.push("Pferd")` hängst du einen Wert hinten an. `push` verändert die Liste und liefert ihre neue Länge zurück. Diese Methode benutzen wir später bei der Browser-App.

Deine Übung

Erstelle ein Array mit vier Farben und gib das komplette Array aus. Verwende für jeden Eintrag einen String.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 17: "Arrays - mehrere Werte unter einem Namen". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: Variablen, Strings, Zahlen und Arrays. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: Ein Array fasst mehrere geordnete Werte zu einer Liste zusammen.

Musterlösung 17: erst nach deinem Versuch prüfen

```
const farben = ["rot", "grün", "blau", "gelb"];
console.log(farben);
```

Lektion 18 Array-Index einzelne Werte finden

Worum geht es

Jedes Element eines Arrays hat eine Position, den sogenannten Index. JavaScript beginnt dabei mit 0.

Warum brauchst du das

Damit kannst du gezielt einen bestimmten Eintrag lesen oder verändern, ohne das ganze Array zu durchlaufen.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
Index	Numerische Position eines Elements im Array.
[0]	Zugriff auf das erste Element.
[1]	Zugriff auf das zweite Element.

Beispiel

```
const tiere = ["Hund", "Katze", "Maus"];
console.log(tiere[1]);
```

Was passiert Schritt für Schritt

- Hund liegt an Index 0.
- Katze liegt an Index 1.
- Maus liegt an Index 2.
- tiere[1] liefert deshalb "Katze".

Deine Übung

Erstelle ein Array mit vier Städten. Gib zuerst das erste und anschließend das vierte Element aus. Schreibe die nötigen Indizes selbst.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 18: "Array-Index - einzelne Werte finden". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: Arrays, Indexzugriff und `console.log`. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: Array-Positionen beginnen bei 0, nicht bei 1.

Musterlösung 18: erst nach deinem Versuch prüfen

```
const staedte = ["Bamberg", "Berlin", "Köln", "Ulm"];
console.log(staedte[0]);
console.log(staedte[3]);
```

Lektion 19 .length die Länge eines Arrays

Worum geht es

`.length` liefert die Anzahl der Elemente eines Arrays.

Warum brauchst du das

Die Anzahl brauchst du für Auswertungen, Schleifen und Grenzprüfungen. Wichtig ist der Unterschied zwischen Länge und letztem Index.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
<code>.length</code>	Eigenschaft mit der Anzahl der Elemente.
letzter Index	Bei einem nicht leeren Array immer <code>length - 1</code> .

Beispiel

```
const farben = ["rot", "grün", "blau", "gelb"];
console.log(farben.length);
console.log(farben[farben.length - 1]);
```

Was passiert Schritt für Schritt

- `farben.length` ist 4.
- Der letzte gültige Index ist deshalb 3.

- `farben.length - 1` ergibt 3.
- `farben[3]` liefert "gelb".

Deine Übung

Erstelle ein Array mit fünf Zahlen. Gib seine Länge und anschließend das letzte Element aus, ohne den letzten Index als feste Zahl einzutippen.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 19: ".length - die Länge eines Arrays". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: Arrays, Indexzugriff, .length und einfache Rechnungen. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: Länge zählt Elemente; Indizes beginnen bei 0.

Musterlösung 19: erst nach deinem Versuch prüfen

```
const zahlen = [3, 6, 9, 12, 15];  
console.log(zahlen.length); // 5  
console.log(zahlen[zahlen.length - 1]); // 15
```

Lektion 20 for...of jedes Element durchlaufen

Worum geht es

`for...of` geht nacheinander durch alle Werte eines Arrays. Für jeden Durchlauf erhältst du den aktuellen Wert in einer Variablen.

Warum brauchst du das

Viele Aufgaben bedeuten schlicht: "Mach mit jedem Eintrag etwas." Dafür ist `for...of` besonders gut lesbar.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
<code>for</code>	Start einer Schleife.
<code>const farbe</code>	Variable für den aktuellen Wert.
<code>of</code>	Sinngemäß: aus dieser Sammlung.
<code>{ ... }</code>	Code, der für jedes Element ausgeführt wird.

Beispiel

```
const farben = ["rot", "grün", "blau"];

for (const farbe of farben) {
  console.log(farbe);
}
```

Was passiert Schritt für Schritt

- Im ersten Durchlauf enthält farbe den Wert "rot".
- console.log gibt rot aus.
- Für den zweiten Durchlauf entsteht eine neue farbe-Bindung mit "grün", danach eine mit "blau".
- Nach dem letzten Element endet die Schleife automatisch.

Warum const in der Schleife funktioniert: Jeder Durchlauf erhält eine eigene Variablenbindung. Die Bindung eines Durchlaufs wird nicht neu zugewiesen. farbe ist ein frei gewählter Name; farben ist der Name deiner vorhandenen Liste. Die Schleife sucht nichts, sondern verarbeitet die Werte in Reihenfolge. Bei einem leeren Array läuft ihr Block kein einziges Mal.

Deine Übung

Erstelle ein Array mit vier Tiernamen. Durchlaufe es mit for...of und gib jeden Namen einzeln aus. Schreibe die Schleife aus dem Gedächtnis.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 20: "for...of - jedes Element durchlaufen". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: Arrays, const, console.log und for...of. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: for...of bedeutet praktisch: für jeden Wert aus diesem Array.

Musterlösung 20: erst nach deinem Versuch prüfen

```
const tiere = ["Hund", "Katze", "Maus", "Pferd"];
for (const tier of tiere) {
  console.log(tier);
}
```

Lektion 21 Mit Schleifen zählen und summieren

Worum geht es

Eine Schleife kann einen Zwischenstand verändern. So lassen sich Werte addieren oder Treffer zählen.

Warum brauchst du das

Dieses Muster steckt hinter sehr vielen Auswertungen: Startwert anlegen, bei jedem Element verändern, Ergebnis danach verwenden.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
Akkumulator	Variable, die ein laufendes Zwischenergebnis sammelt.
Zähler	Variable, die zählt, wie oft etwas passiert.

Beispiel

```
const zahlen = [4, 7, 3];
let summe = 0;

for (const zahl of zahlen) {
  summe = summe + zahl;
}

console.log(summe);
```

Was passiert Schritt für Schritt

- summe beginnt bei 0.
- Nach der 4 ist summe = 4.
- Nach der 7 ist summe = 11.
- Nach der 3 ist summe = 14.
- Erst nach der Schleife wird das Endergebnis ausgegeben.

Deine Übung

Lege [10, 5, 8, 2] an und berechne mit for...of selbst die Summe. Verwende kein reduce und keine fertige Summenfunktion.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 21: "Mit Schleifen zählen und summieren". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: Arrays, for...of, let und Rechnen. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte.

Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: Ein laufendes Ergebnis braucht meist eine `let`-Variable außerhalb der Schleife.

Musterlösung 21: erst nach deinem Versuch prüfen

```
const zahlen = [10, 5, 8, 2];
let summe = 0;
for (const zahl of zahlen) {
    summe = summe + zahl;
}
console.log(summe); // 25
```

Lektion 22 Durchschnitt berechnen

Worum geht es

Für einen Durchschnitt brauchst du die Summe aller Werte und die Anzahl der Werte.

Warum brauchst du das

Die Aufgabe verbindet Array, Schleife, Akkumulator und `.length` zu einem kleinen vollständigen Algorithmus.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
Durchschnitt	Summe aller Werte geteilt durch ihre Anzahl.
leeres Array	Sonderfall mit <code>length 0</code> ; eine Division durch 0 ist fachlich nicht sinnvoll.

Beispiel

```
const zahlen = [10, 20, 30];
let summe = 0;

for (const zahl of zahlen) {
    summe = summe + zahl;
}

if (zahlen.length > 0) {
    const durchschnitt = summe / zahlen.length;
    console.log(durchschnitt);
}
```

Was passiert Schritt für Schritt

- Die Schleife bildet zuerst die Summe 60.

- `zahlen.length` ist 3.
- Die `if`-Abfrage schützt den Sonderfall eines leeren Arrays.
- $60 / 3$ ergibt 20.

Deine Übung

Berechne den Durchschnitt von `[6, 9, 12, 15]`. Prüfe vorher, dass das Array nicht leer ist.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 22: "Durchschnitt berechnen". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: Arrays, Schleifen, `.length`, `if` und Rechnen. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: Durchschnitt = Summe / Anzahl; Sonderfälle gehören zur Lösung dazu.

Musterlösung 22: erst nach deinem Versuch prüfen

```
const zahlen = [6, 9, 12, 15];
let summe = 0;
for (const zahl of zahlen) {
    summe = summe + zahl;
}
if (zahlen.length > 0) {
    console.log(summe / zahlen.length); // 10.5
} else {
    console.log("Keine Werte");
}
```

Lektion 23 Die klassische for-Schleife

Worum geht es

Die klassische `for`-Schleife steuert Start, Bedingung und Veränderung eines Zählers direkt.

Warum brauchst du das

Sie ist nützlich, wenn du den Index brauchst oder nur einen bestimmten Bereich einer Liste durchlaufen willst.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
for (Start; Bedingung; Schritt)	Die drei Steuerteile einer klassischen for-Schleife.
let i = 0	Startwert des Zählers.
i++	Kurzform für i = i + 1.

Beispiel

```
const namen = ["Anna", "Ben", "Clara"];

for (let i = 0; i < namen.length; i++) {
  console.log(i, namen[i]);
}
```

Was passiert Schritt für Schritt

- i startet bei 0.
- Vor jedem Durchlauf wird i < namen.length geprüft.
- namen[i] greift über den aktuellen Index zu.
- Nach jedem Durchlauf erhöht i++ den Zähler um 1.

Deine Übung

Erstelle ein Array mit vier Produkten. Gib mit einer klassischen for-Schleife jeweils Index und Produktname aus.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 23: "Die klassische for-Schleife". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: Arrays, Index, .length, let und klassische for-Schleifen. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: for...of gibt dir direkt den Wert; die klassische for-Schleife gibt dir volle Kontrolle über den Index.

Musterlösung 23: erst nach deinem Versuch prüfen

```
const produkte = ["Tasse", "Buch", "Stift", "Heft"];
for (let i = 0; i < produkte.length; i++) {
  console.log(i, produkte[i]);
}
```

Lektion 24 while, break und continue

Worum geht es

while wiederholt Code, solange eine Bedingung true ist. break beendet eine Schleife sofort; continue springt zum nächsten Durchlauf.

Warum brauchst du das

Diese Werkzeuge helfen, wenn die Anzahl der Wiederholungen nicht vorher feststeht oder du Schleifen gezielt steuern musst.

Neue Begriffe, Befehle und Zeichen

Begriff / Zeichen	Bedeutung
while	Wiederhole, solange die Bedingung true ist.
break	Beendet die aktuelle Schleife sofort.
continue	Überspringt den Rest des aktuellen Durchlaufs.

Beispiel

```
let zahl = 0;

while (zahl < 5) {
  zahl++;
  if (zahl === 2) {
    continue;
  }
  if (zahl === 4) {
    break;
  }
  console.log(zahl);
}
```

Was passiert Schritt für Schritt

- zahl startet bei 0 und wird im while-Block erhöht.
- Bei 2 überspringt continue die Ausgabe.
- Bei 3 wird ausgegeben.
- Bei 4 beendet break die Schleife vollständig.

Die Ausgabe ist 1 und 3. continue überspringt den Rest des Blocks, nicht die nächste Zahl automatisch. Darum muss der Zähler vor continue erhöht werden. Wenn du eine Endlosschleife erzeugst, stoppe den laufenden Prozess im Terminal mit Strg+C; bei Code Runner nutze dessen Stopp-Befehl.

Deine Übung

Schreibe eine while-Schleife, die bei 1 startet und Zahlen bis 5 zählt. Überspringe die 3 mit continue und beende die Schleife spätestens bei 5.

Mit KI weiterüben

Ich lerne JavaScript und bin bei Lektion 24: "while, break und continue". Stelle mir nacheinander drei kurze Übungen ausschließlich mit: Schleifen, Bedingungen, break und continue. Verwende keine späteren Konzepte und zeige keine fertige Lösung. Bei Fehlern weise zuerst nur darauf hin, danach gib einen kleinen Hinweis. Die vollständige Lösung erst, wenn ich ausdrücklich darum bitte. Ich werde dir bei Bedarf die Lektion oder meine Liste erlaubter Befehle mitgeben. Frage nach, wenn ein Begriff unklar ist; führe keine unbekannten Hilfsmethoden stillschweigend ein. Warte nach jeder einzelnen Aufgabe auf meinen Versuch.

Merksatz: Eine while-Bedingung muss sich irgendwann ändern können, sonst droht eine Endlosschleife.

Musterlösung 24: erst nach deinem Versuch prüfen

```
let zahl = 1;
while (zahl <= 5) {
  if (zahl === 3) {
    zahl++;
    continue;
  }
  console.log(zahl);
  if (zahl === 5) {
    break;
  }
  zahl++;
}
// Ausgabe: 1, 2, 4, 5
```

Dein Abschlusscheck

Du hast den Einstieg bis Arrays und Schleifen bearbeitet. Jetzt verbindest du mehrere bekannte Grundlagen in einem kleinen Einkaufsprogramm. Schau dir die Musterlösung auf Seite 56 erst nach deinem eigenen Versuch an.

Aufgabe: Passt der Einkauf ins Budget?

Beginne mit einer leeren Übungsdatei. Lege ein Array `preise` mit den Werten `[8, 12, 5]` und eine Variable `budget` mit dem Wert 25 an.

1. Gib die Anzahl der Preise aus.
2. Berechne die Gesamtsumme mit `for...of` und einer `let`-Variable. Verwende keine Funktion und keine fertige Summenmethode.
3. Gib die Summe aus. Prüfe anschließend mit `if/else`, ob der Einkauf ins Budget passt. Gleichheit soll als „passt“ gelten.
4. Gib eine klare Meldung aus. Wenn das Budget reicht, zeige auch den Restbetrag; andernfalls den fehlenden Betrag.

Prüfe mehrere Fälle

Teste denselben Einkauf mit Budget 25, 24 und 30. Teste danach ein leeres Array bei Budget 25. Für diese Aufgabe gilt: Ein leerer Einkauf kostet 0 und passt ins Budget. Verwende nur nicht negative Preise und Budgets.

Erkläre deinen eigenen Code

Warum beginnt die Summe bei 0? Warum steht sie außerhalb der Schleife? Wofür brauchst du `let`? Was passiert bei einem leeren Array? An welcher Stelle läuft die Budgetprüfung?

Prüfen, wiederholen, weiterlernen

Was die Testfälle zeigen sollen

Bei [8, 12, 5] beträgt die Summe 25. Budget 25 reicht genau; bei 24 fehlt 1; bei 30 bleiben 5. Bei [] ist die Anzahl 0, die Summe 0 und bei Budget 25 bleiben 25. Vergleiche Ausgabe und Ablauf mit diesen Erwartungen.

Deinen Lernstand ehrlich festhalten

Selbst geschrieben: Du konntest den Ablauf ohne fertigen Code aufbauen und erklären.

Mit Unterstützung gelöst: Du hast nachgeschlagen oder Hinweise genutzt. Notiere, an welcher Stelle.

Noch unsicher: Eine Stelle funktioniert nur nach Vorlage oder lässt sich noch nicht erklären. Wiederhole dazu die passende Lektion.

KI-Mischtraining für diese Leseprobe

Ich habe die ersten 24 Lektionen von JavaScript von Null bis Praxis bearbeitet. Erlaubt sind Werte, const/let, Rechnen, Vergleiche, if/else/else if, logische Operatoren, Arrays, Indexzugriff, .length, for...of, klassische for-Schleifen, while, break und continue. Stelle mir sechs gemischte Aufgaben, immer nur eine gleichzeitig. Beginne leicht und steigere die Kombination bekannter Inhalte. Gib keine fertige Lösung, bevor ich ausdrücklich darum bitte. Lass mich Fehler zunächst selbst suchen und gib danach Hinweise stufenweise. Verwende keine Funktionen, Objekte oder unbekannten Methoden. Prüfe auch, ob ich meinen Code erklären und verändern kann. Fasse anschließend beobachtete Stärken und Unsicherheiten zusammen und stelle gezielte Wiederholungsaufgaben. Kennzeichne nicht geprüfte Themen als offen. Warte jetzt auf meinen ersten Versuch.

Die Einschätzung der KI ist eine Lernhilfe und kann falsch sein. Dein eigener Code, seine Ausgabe und deine Erklärung sind die Grundlage. Ein kurzer Abschlusscheck belegt noch keine langfristige Sicherheit.

Musterlösung: erst jetzt vergleichen

Dies ist eine mögliche Umsetzung. Andere Lösungen sind ebenso richtig, wenn sie die Aufgabe erfüllen. Nutze zum Prüfen auch die Testfälle von Seite 55.

```
const preise = [8, 12, 5];
const budget = 25;
let summe = 0;

for (const preis of preise) {
  summe = summe + preis;
}

console.log("Anzahl:", preise.length);
console.log("Summe:", summe);

if (summe <= budget) {
  console.log("Der Einkauf passt ins Budget.");
  console.log("Restbetrag:", budget - summe);
} else {
  console.log("Das Budget reicht nicht.");
  console.log("Fehlender Betrag:", summe - budget);
}
```

Warum diese Lösung funktioniert

Die Schleife addiert jeden Preis zur laufenden Summe. Bei einem leeren Array wird ihr Block nicht ausgeführt; summe bleibt 0. Erst danach wird das Ergebnis mit dem Budget verglichen. Die Bedingung `summe <= budget` schließt den Fall ein, dass das Budget genau reicht.

Schließe die Musterlösung und ändere die Aufgabe: Prüfe nun zusätzlich, ob mehr als vier Produkte im Einkauf liegen. Schreibe diese kleine Erweiterung selbst mit den bereits gelernten Mitteln.

So geht dein Lernweg weiter

Mit dieser Leseprobe hast du den Lernweg vom ersten Programm bis zu Arrays und Schleifen kennengelernt. Wenn du die Aufgaben selbst schreiben, erklären und verändern kannst, hast du eine Grundlage für die nächsten Schritte. Falls etwas noch unsicher ist, lohnt sich zuerst eine Wiederholung.

Im vollständigen Buch folgen unter anderem

Funktionen: Abläufe wiederverwenden, Werte übergeben und Ergebnisse zurückgeben.

Objekte: zusammengehörige Daten strukturieren und verarbeiten.

Dateien, JSON und Schnittstellen: Daten lesen, speichern und von Diensten abrufen.

Browseroberflächen: auf Eingaben und Klicks reagieren.

Fehlersuche, Tests und Git: Programme nachvollziehbar prüfen und weiterentwickeln.

Der Grundlagenband umfasst 104 Lektionen mit Übungen und KI-Lernprompts. Er führt zu ersten praktischen Anwendungen; er deckt nicht sämtliche JavaScript-Themen ab.

Das vollständige Buch

JavaScript von Null bis Praxis

104 Lektionen für Anfänger – Grundlagen verstehen, selbst programmieren und mit KI üben

Johann Eitlhuber

ISBN des Taschenbuchs: 9798160345604

Die Veröffentlichung wird derzeit vorbereitet. Sobald das Buch erhältlich ist, kannst du auf Amazon nach Titel und Autor suchen und die verfügbaren Ausgaben prüfen.

Fragen oder Hinweise zur Leseprobe?

Joale689@gmail.com