IN DIESEM KAPITEL...

- Erste JavaScript-Befehle im Browser ausprobieren
- Fremde Webseiten (vorübergehend) verändern
- Überblick über Programmiersprachen
- Mehrzeilige Mini-Programme

Kapitel 1
Erste Woche: Erste Schritte

In dieser Woche führen Sie erste JavaScript-Befehle aus, verändern testweise Webseiten im Browser und bekommen ein Gefühl für grundlegende Konzepte wie Variablen, Ausgabe und einfache Operationen. Außerdem erhalten Sie einen Überblick über verbreitete Programmiersprachen und festigen das Gelernte mit einem kurzen Quiz.

Es sind keine Vorkenntnisse nötig. Grundlegende Computerbedienung und Neugier reichen aus.

Tag 1: Worauf Sie sich freuen können: Ihr eigenes Spiel programmieren

Wenn Sie das Buch durchgearbeitet haben, dann können Sie ein Spiel programmieren: ein richtiges klassisches Weltraumspiel. Mit Sternen, einem Raumschiff, das Sie steuern dürfen, und einem Punktestand mit Highscore und sogar einer Laserkanone. In Abbildung 1.1 sehen Sie, wie das Spiel am Ende aussehen wird. Es ist vielleicht nicht das schönste Spiel auf der Welt, aber es ist selbst gemacht. Und das ist doch schon mal was!

Programmieren macht mir großen Spaß. So sehr, dass ich es nicht nur beruflich mache, sondern auch in meiner Freizeit. Während der Corona-Pandemie habe ich zum Spaß einmal ein Online-Kartenspiel programmiert, das ich über das Internet mit meinen Freunden spielen konnte. Solche Browsergames gab es natürlich zuhauf im Internet, aber ich wollte es selbst machen – einfach nur, weil ich es kann. Und weil es Spaß macht.

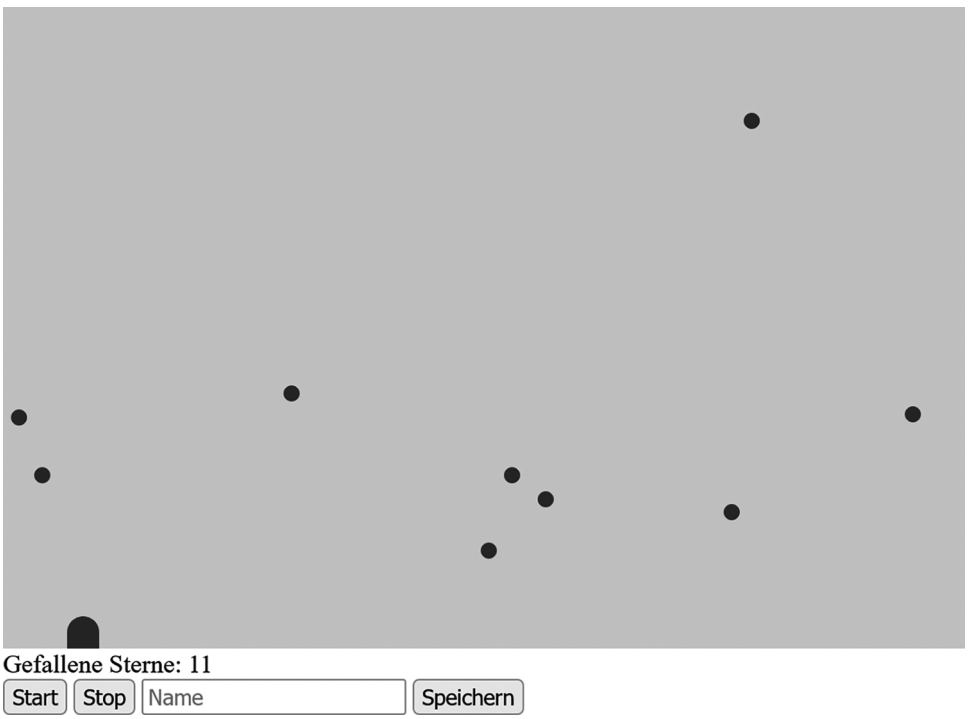

Abbildung 1.1: Wie Sie dieses Spiel selbst programmieren können, erfahren Sie in diesem Buch. Es ist vielleicht nicht das schönste Spiel auf der Welt, aber es ist selbst gemacht!

Was ist Programmieren?

Programmieren besteht im Wesentlichen daraus, Programmcode zu schreiben – und zwar in einer bestimmten Programmiersprache, so wie Sie es in Abbildung 1.2 sehen. Der Code selbst ist eine Folge von Anweisungen, die der Computer ausführen kann – wie ein Kochrezept, Schritt für Schritt. Der Code ist für Menschen lesbar; die Fachausdrücke (auch *Schlüsselwörter* genannt) sind allerdings in der Regel in englischer Sprache. Bevor der Code vom Computer ausgeführt wird, wird er in Maschinencode übersetzt. Dieser Prozess läuft meist im Hintergrund, als Programmiererin oder Programmierer hat man damit normalerweise wenig bis nichts zu tun.

Abbildung 1.2: Beim Programmieren schreiben Sie Code in einer Programmiersprache, die aus einer Abfolge von Anweisungen besteht. Der Code wird dann automatisch in Maschinencode übersetzt, den der Computer versteht.

Pseudocode

Was braucht es, um ein einfaches Spiel zu programmieren? Nehmen wir an, Sie wollen ein Weltraumspiel programmieren. Ein Raumschiff soll sich mit den Pfeiltasten nach links und rechts bewegen. Sobald das Raumschiff von einem Stern getroffen wird, ist das Spiel vorbei.

Wir brauchen also einen Mechanismus, der die Tastatur abfragt, einen zweiten Mechanismus, der ein Raumschiff auf dem Bildschirm anzeigt, und eine Steuerung, die das Raumschiff bewegt, sobald eine Pfeiltaste gedrückt wird.

Der Ablauf lässt sich – stark vereinfacht – so beschreiben:

```
Zeichne das Bild Raumschiff.jpg in die Mitte des Bildschirms
Wenn die Taste Pfeil nach links gedrückt wird
   Bewege das Raumschiff nach links
Wenn die Taste Pfeil nach rechts gedrückt wird
   Bewege das Raumschiff nach rechts
```

```
Haben ein Stern und das Raumschiff die gleiche Position?
   Dann ist das Spiel vorbei
```

Aus diesem einfachen Ablauf lässt sich bereits ein formales Programm ableiten:

```
Zeichne(Raumschiff.jpg, Mitte_des_Bildschirms)

Wiederhole:
   Wenn TasteLinksGedrückt():
       Raumschiff.BewegeNachLinks()

   Wenn TasteRechtsGedrückt():
       Raumschiff.BewegeNachRechts()

   Wenn Position(Raumschiff) == Position(Stern):
       SpielEnde()
```

Was Sie hier sehen, ist sogenannter *Pseudocode*. Er ähnelt einer Programmiersprache, funktioniert aber nicht real, sondern verdeutlicht nur die Idee hinter dem Programm. Zunächst wird das Bild des Raumschiffs in die Bildschirmmitte gezeichnet. Dann wird abgefragt, ob die Pfeiltaste nach links oder die nach rechts gedrückt werden. Wenn ja, wird das Raumschiff entsprechend bewegt. Das ist alles, was wir brauchen.

Pseudocode ist eine vereinfachte, sprachliche Darstellung eines Algorithmus. Er ist nicht formal an eine konkrete Programmiersprache gebunden und hilft dabei, Abläufe zu skizzieren und zu kommunizieren.

Natürlich ist das nur ein sehr grober Überblick. In Wirklichkeit gibt es viele weitere Fragen zu beantworten:

- ✔ Wie zeigen wir das Raumschiff auf dem Bildschirm an?
- ✔ Wie bewegen wir das Raumschiff?
- ✔ Wie erkennen wir, dass eine Pfeiltaste gedrückt wird?
- ✔ Was passiert, wenn das Raumschiff am Rand des Bildschirms ist?
- ✔ Wie berechnen wir den Punktestand und zeigen ihn an?
- ✔ Wie speichern wir den Punktestand?
- ✔ und noch vieles mehr

Sie sehen: In der Programmierung gibt es viele Fragen zu beantworten – mit wachsendem Detailgrad. Die grundlegenden Fragen, die man sich bei der Lösung einer Programmieraufgabe stellt, unterscheiden sich zwischen den Programmiersprachen kaum. Auch eine moderne Kaffeemaschine muss feststellen: Soll ich brühen? Muss ich entkalkt werden ? Wie zeige ich das an? Und was mache ich, wenn keiner reagiert?

Programmieren ist dem Schreiben eines Kochrezepts ähnlich. Man benötigt eine unmissverständliche Anleitung, um ein bestimmtes Ergebnis zu erzielen.

Die Programmierung einer Kaffeemaschine hat tatsächlich mehr mit einem Weltraumspiel gemeinsam, als Sie vielleicht denken. Und das ist auch der Grund, warum ich Ihnen dieses Buch empfehle: Es ist nicht nur ein Buch über JavaScript, sondern auch eines über das Programmieren an sich.

Allen Programmiersprachen ist gemeinsam, dass ihre Programme aus einer Reihe von Anweisungen bestehen, die der Computer ausführen kann. Der Computer folgt den Anweisungen Schritt für Schritt, um das gewünschte Ergebnis zu erzielen. Und wenn das Ergebnis nicht das gewünschte ist, dann haben Sie vermutlich das Rezept falsch beschrieben oder irgendwo einen Denkfehler gemacht. Der Computer macht genau das, was man ihm sagt – auch wenn es falsch ist.

JavaScript

Um richtig programmieren zu können, brauchen wir eine Programmiersprache – Pseudocode allein reicht nicht. Es gibt unzählige Programmiersprachen, die jeweils einen bestimmten Zweck erfüllen: Vielleicht kennen Sie *Python*, das in der Datenanalyse verwendet wird, oder *Java* oder *C*. Letzteres verwendet vielleicht Ihre Kaffeemaschine, um zu erkennen, dass sie entkalkt werden muss.

In diesem Buch arbeiten wir mit JavaScript. Das ist eine der am häufigsten verwendeten Programmiersprachen im Internet. Sie wird genutzt, um Webseiten interaktiv zu machen und um Spiele zu programmieren. JavaScript ist vergleichsweise leicht zu lernen und bietet viele Funktionen, mit denen sich selbst komplexe Spiele erstellen lassen.

Das Vorgehen bei der Programmierung ist in jeder Programmiersprache ziemlich ähnlich. Zuerst versucht man, das Problem zu verstehen und zu beschreiben. Anschließend zerlegt man die Lösung in Teilschritte. Dabei kann Pseudocode sehr hilfreich sein. Erst wenn diese Teilschritte klar sind, wird der eigentliche Code geschrieben.

First solve the problem,
then write the code.

Die allerersten Schritte mit JavaScript sind ganz einfach – morgen schon fangen wir damit an.

Tag 2: Erste JavaScript-Befehle direkt im Browser

Heute fangen wir direkt mit dem Programmieren an. Sie erstellen in wenigen Minuten ein kleines Programm. Ziel ist es, auf dem Bildschirm ein kleines Fenster mit den Worten »Hallo Welt!« anzuzeigen.

Dafür brauchen wir einen Browser, zum Beispiel *Chrome*, *Firefox* oder *Safari*. Ich verwende *Firefox*, aber Sie können auch jeden anderen Browser benutzen. Im Browser rufe ich die Seite `https://www.wiley-vch.de/de/dummies` auf; Sie können aber auch jede andere Webseite nehmen.

Um programmieren zu können, müssen wir in den sogenannten *Entwicklermodus* wechseln. Das ist ein spezieller Modus, der es Ihnen ermöglicht, den Code einer Webseite zu inspizieren, zu bearbeiten und zu testen.

In Abbildung 1.3 sehen Sie die Ansicht für Firefox. Dort öffnen Sie die Entwicklertools, indem Sie die F12 drücken oder mit der rechten Maustaste auf eine beliebige Stelle der Seite klicken und Untersuchen wählen. In Chrome und Edge ist es ähnlich: Rechtsklick und Untersuchen. Alternativ können Sie in allen Browsern die Tastenkombination Strg ⇧ I verwenden bzw. ⌘ ⌥ I auf dem Mac.

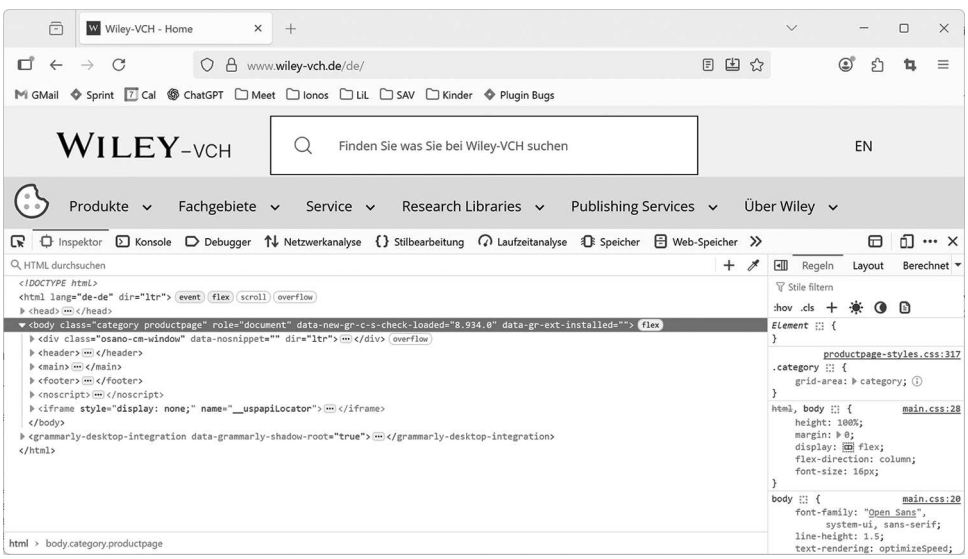

Abbildung 1.3: Der Entwicklermodus in Firefox. Unten sehen Sie die Konsole für JavaScript-Befehle. Sie öffnen sie mit F12 oder der Tastenkombination Strg ⇧ I.

In Safari müssen Sie die Entwicklertools zuerst aktivieren: In den Einstellungen wechseln Sie in den Reiter Erweitert und aktivieren die Option Menü 'Entwickler' in der Menüleiste anzeigen. Danach können Sie den Entwicklermodus über F12 oder Element untersuchen öffnen.

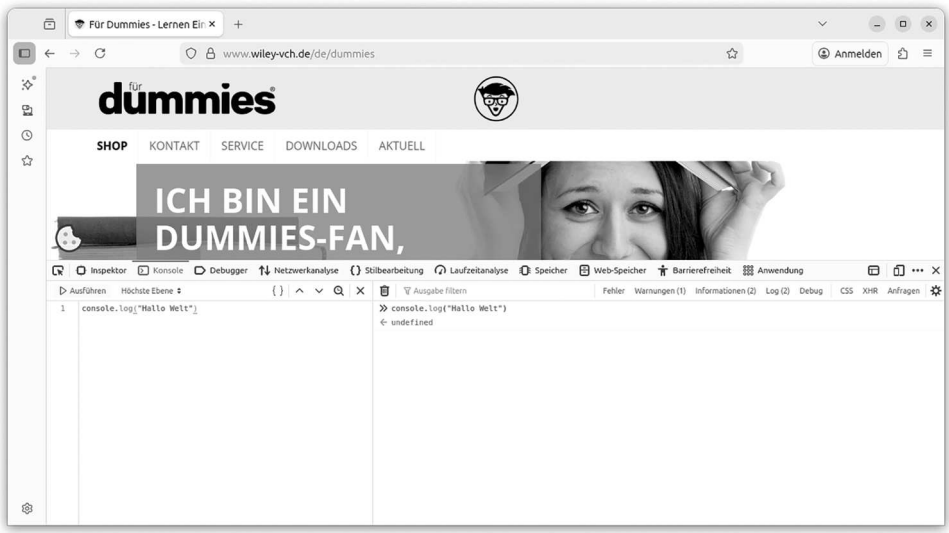

Abbildung 1.4: Entwicklermodus mit geöffneter Konsole.

Wenn Ihr Fenster anders als in Abbildung 1.4 aussieht, ist das in Ordnung. Wichtig ist nur, dass Sie die Konsole sehen. Falls nicht, klicken Sie auf den Tab Konsole bzw. Console.

Hallo Welt!

An dieser Konsole können Sie nun JavaScript-Befehle eingeben und sie ausführen lassen – das ist fast schon programmieren! Geben Sie den folgenden Befehl ein und drücken Sie in Chrome Return oder in Firefox Strg Return, um ihn auszuführen:

```
alert('Hallo Welt!')
```

Sie sollten jetzt ein kleines Fenster wie in Abbildung 1.5 sehen, das »Hallo Welt!« anzeigt. Das ist Ihr erster erfolgreich ausgeführter JavaScript-Befehl. Glückwunsch – Ihr erstes Programm läuft.

Im Entwicklermodus sehen Sie auch den HTML-Code der Seite: In Chrome heißt der Reiter Elements, in Firefox Inspektor. Ein weiterer Reiter heißt Konsole. Dort führen Sie JavaScript aus. Mit dem Befehl »`alert('Hallo Welt!');`« erscheint unser Begrüßungsfenster.

Wir können im Entwicklermodus noch viel mehr anstellen – zum Beispiel die komplette Seite verschwinden lassen. Tippen Sie:

```
document.body.innerHTML = "Hallo Welt!";
```

Die Webseite im Hintergrund ist weg und nur »Hallo Welt!« bleibt. Laden Sie die Seite mit F5 oder Neu laden neu – alles ist wieder da.

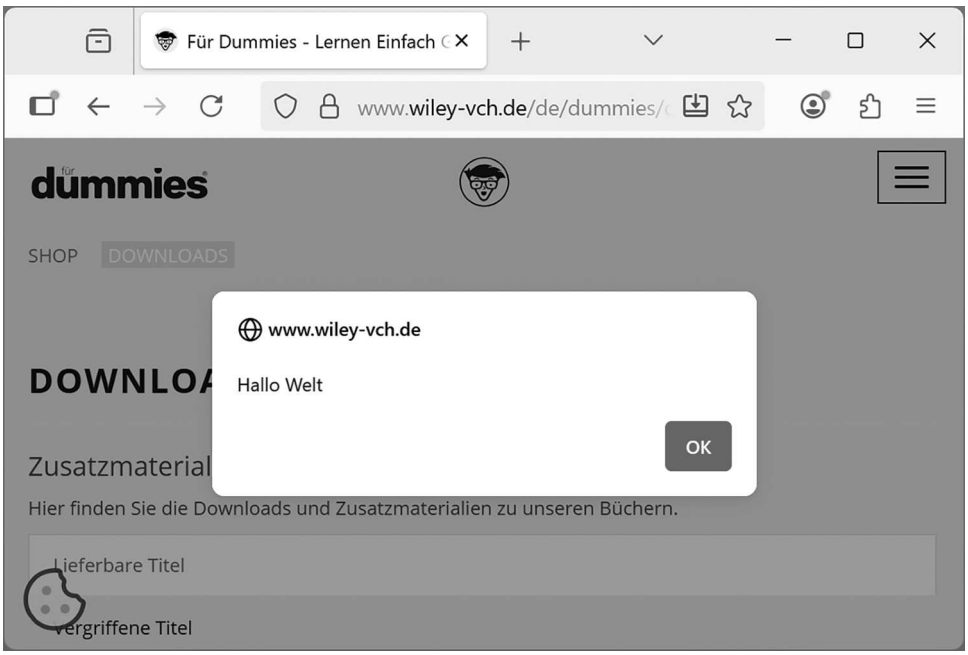

Abbildung 1.5: Das Ergebnis des Befehls `alert('Hallo Welt!')`.

So ist ein Befehl aufgebaut

Der erste Befehl, den wir kennengelernt haben, war `alert('Hallo Welt!');`. Er öffnet ein kleines Dialogfenster mit Text.

So ist er aufgebaut: Erst der Funktionsname `alert`, dann runde Klammern mit dem Text in Anführungszeichen. Der Befehl endet mit einem Semikolon (Strichpunkt), das ist aber nicht immer zwingend notwendig. In JavaScript ist das Semikolon häufig optional, aber konsequente Nutzung macht Ihren Code klarer, was gerade am Anfang sehr ratsam ist.

Wichtig sind:

- ✔ exakte Schreibweise von `alert`
- ✔ runde Klammern
- ✔ passende Anführungszeichen (einfach oder doppelt, aber als Paar)

Probieren Sie Varianten:

- ✔ Ersetzen Sie einfache durch doppelte Anführungszeichen.
- ✔ Lassen Sie die Klammern weg.
- ✔ Entfernen Sie das Semikolon.

Sollten hier Fehler passieren, sind sie harmlos; mit F5 setzen Sie alles zurück.

Der zweite Befehl `document.body.innerHTML = "Hallo Welt!";` weist einer Eigenschaft einen neuen Wert zu. Links steht, was geändert wird, und zwar innerhalb des DOCUMENT den eigentlichen Inhalt BODY und darin INNERHTML; rechts nach dem Gleichheitszeichen folgt der neue Inhalt.

Tippen Sie zum Beispiel:

```
document.body.style.background = "red";
```

Scrollen Sie nach unten: Die Hintergrundfarbe ist jetzt rot. Probieren Sie `blue`, `green`, `yellow` aus.

Die Funktionalität Autovervollständigen hilft hier: Tippen Sie

```
document.body.style.c
```

und wählen Sie einen Vorschlag, um zum Beispiel die Textfarbe (`color`) zu ändern.

Tag 3: Mehr JavaScript-Befehle im Browser

Gestern haben Sie die ersten Befehle in JavaScript ausprobiert – das waren kleine, einzeilige Programme. Heute schreiben wir etwas komplexere Programme mit mehreren Zeilen.

Wir verwenden dazu wieder den Entwicklermodus im Browser. Öffnen Sie Ihren Browser und drücken Sie die Taste F12 oder die Tasten Strg ⇧ I, um die Entwicklertools zu öffnen. Dort klicken Sie auf den Tab Konsole.

Schauen Sie sich das folgende kleine Programm an und überlegen Sie, was es macht:

```
name = prompt("Wie heißen Sie?")
document.body.innerHTML = "Hallo " + name
```

Haben Sie es erraten, wofür der Programmcode steht und was er tut? Geben Sie den Code in die Konsole ein und führen Sie ihn mit der Eingabetaste aus. Das Programm fragt Sie erst nach Ihrem Namen und gibt Ihnen dann eine persönliche Begrüßung zurück.

Das Programm besteht ganz offensichtlich aus zwei Zeilen: In der ersten nutzen wir eine sogenannte Variable namens `name`. Darin speichern wir den Namen des Nutzers oder der Nutzerin.

Wie kommen wir an den Namen? Wir fragen einfach danach. Mit dem Schlüsselwort `prompt` erzeugen wir eine Eingabebox. Ein kleines Fenster öffnet sich, in dem man den Namen eingeben kann. Das Ergebnis wird in der Variablen `name` gespeichert.

Eine Variable ist ein Platzhalter für einen Wert. In diesem Fall speichern wir den abgefragten Namen in `name`. Sie können den Inhalt einer Variablen später abrufen oder auch ändern – wie bei einer Schublade.

In der zweiten Zeile geben wir Text aus, nämlich erst »Hallo« und dann den Inhalt der Variablen `name`. Achten Sie auf die Anführungszeichen und das Pluszeichen! Alles in Anführungszeichen wird genauso ausgegeben, wie es geschrieben wurde.

Beim Namen wollen wir jedoch natürlich den eingegebenen Namen und nicht das Wort »Name« ausgeben. Deshalb steht die Variable nicht in Anführungszeichen. Das Pluszeichen verbindet beide Textteile.

Hier könnten Ihnen jetzt vielleicht ein paar Bedenken kommen:

- ✔ Was passiert, wenn Sie das Pluszeichen weglassen?
- ✔ Was passiert, wenn Sie den Strichpunkt (falls vorhanden) weglassen?
- ✔ Können Sie die Anführungszeichen weglassen?
- ✔ Wo sind zusätzliche Leerzeichen egal, wo nicht?

Auch hier gilt: Kaputt machen können Sie nichts, der Browser weist Sie wie in Abbildung 1.6 rechtzeitig auf Fehler hin. Um neu zu starten, drücken Sie die Taste F5.

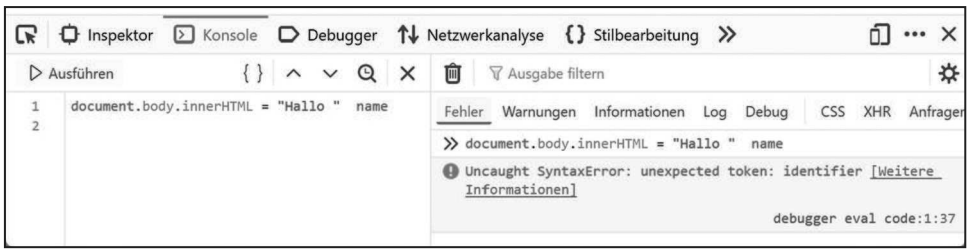

Abbildung 1.6: Der Browser weist deutlich auf einen Fehler im Programmcode hin (rechts unten).

Berechnungen durchführen

Als Nächstes wollen wir etwas mehr wagen und eine Berechnung durchführen. Wir fragen den Nutzer oder die Nutzerin nach einer Zahl, multiplizieren sie mit sich selbst (quadrieren sie) und zeigen das Ergebnis an.

Skizzieren Sie zuerst den Ablauf (Pseudocode):

```
Frage den Nutzer nach einer Zahl (prompt)
Speichere die Zahl in einer Variablen
Multipliziere die Zahl mit sich selbst
Gib das Ergebnis in einer Box aus
```

Die Umsetzung in JavaScript:

```
number = prompt("Geben Sie eine Zahl ein")
square = number * number
document.body.innerHTML = "Quadratzahl von " + number + ": " + square
```

Das Programm fragt nach einer Zahl, speichert sie in `number`, berechnet das Quadrat und gibt das Ergebnis aus.

In diesem Beispiel sehen Sie eine bunte Mischung aus Deutsch und Englisch. Dabei gilt: Alles, was zwischen Anführungszeichen steht, erscheint genau so auf dem Bildschirm (hier Deutsch). Variablennamen sind Englisch – das ist üblich, aber nicht verpflichtend.

Englisch ist die gemeinsame Sprache der Programmierung. Englische Namen für Variablen und Funktionen machen es leichter, den Code in internationalen Teams zu verstehen und Hilfe in Online-Foren zu finden.

Code verbessern

In der Programmierung gibt es oft mehrere Wege zum Ziel. Manchmal ist ein anderer Weg kürzer und klarer. Die Variable `square` ist beispielsweise nicht zwingend nötig. Wir können stattdessen auch direkt im Ausgabestring rechnen:

```
number = prompt("Geben Sie eine Zahl ein")
document.body.innerHTML =
  "Das Quadrat von " + number + " ist " + number * number
```

Dieser Code ist kürzer und leichter zu lesen, das Ergebnis dasselbe. Häufig führen mehrere Wege zum Ziel.

Refactoring bezeichnet das Umschreiben von Code zur Verbesserung von Lesbarkeit, Struktur oder Wartbarkeit – ohne das Verhalten zu ändern. Das verringert Redundanzen und erhöht die Klarheit.

Tag 4: Fremde Webseiten anpassen und verändern

Bisher haben wir mit JavaScript nur in der Konsole experimentiert. Jetzt wollen wir eine vorhandene Webseite gezielt verändern – damit kann man im Freundeskreis schnell Eindruck schinden (natürlich ändern wir das nur für uns, das Netz bekommt davon nichts mit).

HTML: Quelltext einer Webseite

Schauen wir uns den Quelltext einer beliebigen Seite an. Ich besuche hierzu die Seite `https://www.bundesliga.com`. Wenn Sie dort Strg U drücken oder mit der rechten Maustaste auf die Seite klicken und SEITENQUELLTEXT ANZEIGEN wählen, sehen Sie den HTML-Code der Seite wie in Abbildung 1.7. Das sieht kryptischer aus, als es tatsächlich ist.

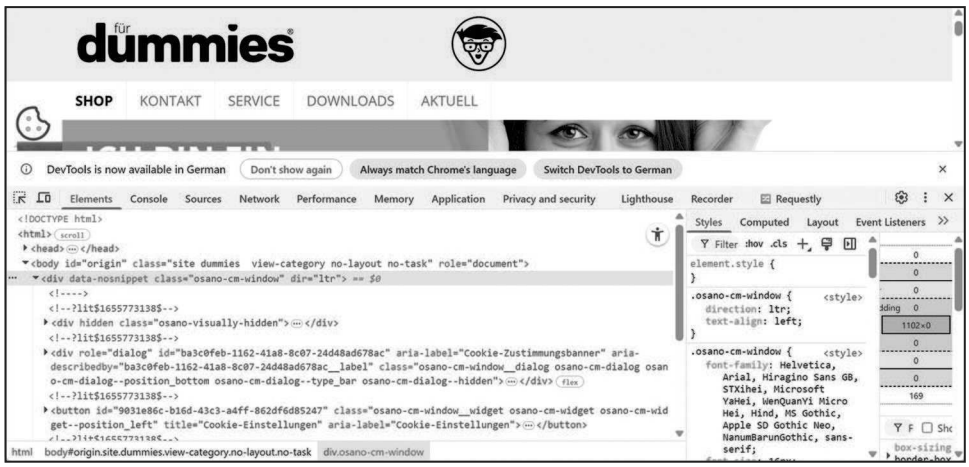

Abbildung 1.7: Der Quelltext einer Webseite besteht aus HTML-Tags, die in spitzen Klammern stehen.

Denselben Quelltext sehen Sie auch in den Entwicklertools, so wie in Abbildung 1.8 gezeigt. Klicken Sie dazu auf den Tab ELEMENTE bzw. INSPEKTOR. Dort sehen Sie den HTML-Code der Seite. Wenn Sie auf einen bestimmten Teil der Seite mit der rechten Maustaste klicken und dann UNTERSUCHEN wählen, wird der entsprechende HTML-Code hervorgehoben. Das ist praktisch, wenn Sie wissen wollen, wie eine bestimmte Stelle im Quelltext aufgebaut wird. Der Quelltext besteht aus HTML-Tags in spitzen Klammern. Diese Tags beschreiben die Struktur der Seite. Zum Beispiel steht `<h1>` für eine Überschrift erster Ebene, `<p>` für einen Absatz und `<a>` für einen Link.

Eine Webseite verändern

Jetzt ändern wir eine Überschrift auf `bundesliga.com`. Aus »Die Rückkehr des Kapitäns« machen wir »Thomas Rose ist Fußballgott«. Natürlich sieht die originale Seite bei Ihnen

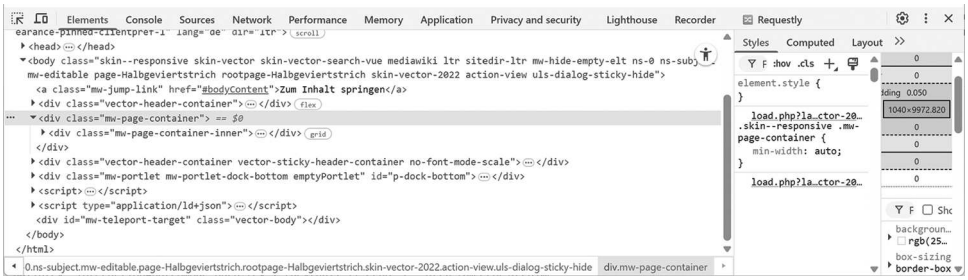

Abbildung 1.8: Der Entwicklermodus im Browser. Nach einem Rechtsklick auf ein Element und der Wahl von UNTERSUCHEN wird der entsprechende HTML-Code hervorgehoben.

vermutlich anders aus – die Inhalte ändern sich laufend. Klicken Sie mit der rechten Maustaste auf eine Überschrift und wählen Sie UNTERSUCHEN. Dann sehen Sie in den Entwicklertools den HTML-Code, der für diese Überschrift verantwortlich ist, so wie in Abbildung 1.9 gezeigt.

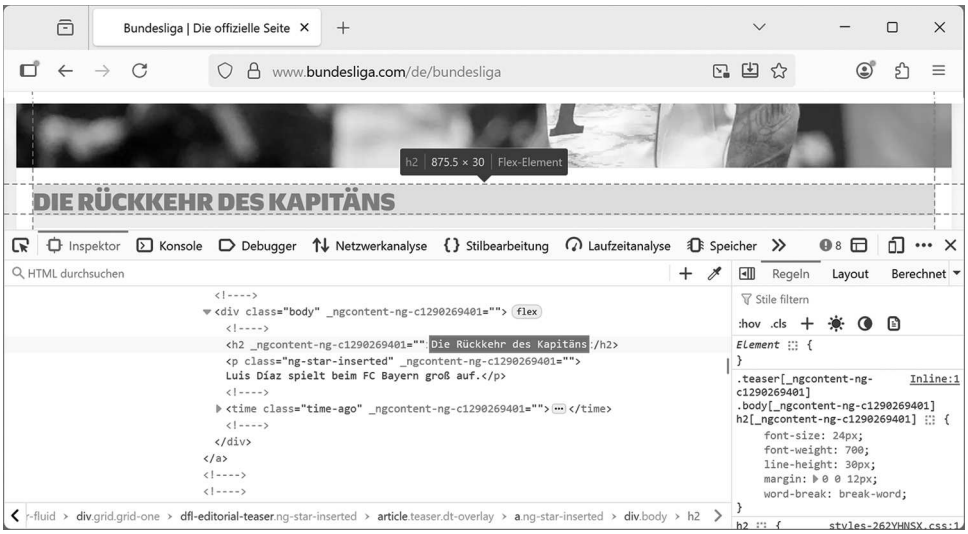

Abbildung 1.9: Der Inspektor zeigt den HTML-Code der Seite.

In den Entwicklertools klicken Sie doppelt auf den Text, ändern ihn in »Thomas Rose ist Fußballgott« und drücken die Eingabetaste. Voilà – die Überschrift ist geändert.

Jetzt können Sie einen Screenshot machen, auf den 1. April warten und das Bild Ihren Freundinnen und Freunden schicken.

Probieren Sie herum: Sie können jeden Text, jedes Bild und viele andere Elemente verändern.

Haben Sie kalte Füße bekommen? Keine Angst – wenn Sie die Seite neu laden, ist alles wieder beim Alten. Sie haben nur die Darstellung in Ihrem *lokalen* Browser verändert. Die echte Webseite bleibt unverändert. Beim Neuladen wird der Originalcode erneut vom Server geladen – Ihre Änderungen sind verschwunden.

Tag 5: Welche Programmiersprachen gibt es?

Es gibt unzählige Programmiersprachen, die jeweils einen bestimmten Zweck erfüllen. Einige bekannte Beispiele:

- ✔ **Python**: Beliebt für Datenanalyse, Machine Learning und Webentwicklung; einfache Syntax, viele Bibliotheken.
- ✔ **Java**: Weit verbreitet für Unternehmensanwendungen und Android. Java ist plattformunabhängig, Code kann also auf dem Mac, PC und Linux laufen.
- ✔ **JavaScript**: Läuft in jedem modernen Browser; mit Frameworks wie React, Angular oder Vue.js für interaktive Webanwendungen.
- ✔ **C/C++**: Systemnahe Programmierung und Spieleentwicklung; hohe Leistung, direkte Speicher- und Hardwarekontrolle, aber komplex.
- ✔ **Ruby**: Fokus auf Einfachheit und Lesbarkeit; durch Ruby on Rails in der Webentwicklung populär.
- ✔ **PHP**: Serverseitig für Webanwendungen; Basis vieler Content-Management-Systeme (CMS) wie WordPress, Joomla oder Drupal.
- ✔ **Swift**: Für iOS- und macOS-Entwicklung; moderne Syntax, sicher und performant.
- ✔ **Go**: Von Google; effizient, einfach, gut für Cloud-Dienste und parallele Anwendungen.
- ✔ **Rust**: Sichere und schnelle Systemprogrammiersprache; verhindert viele Speicherfehler schon zur Compile-Zeit.
- ✔ **Kotlin**: Moderne Sprache für Android; interoperabel mit Java.
- ✔ **R**: Häufig eingesetzt in Statistik und Datenanalyse; viele Pakete für Visualisierung und wissenschaftliches Rechnen.

JavaScript und Java haben kaum Gemeinsamkeiten. JavaScript hieß ursprünglich *LiveScript*; der Name wurde aus Marketinggründen geändert.

Programmiersprachen lassen sich grob einteilen:

- ✔ **Hochsprachen**: Leicht verständlich, abstrahieren technische Details (Python, Java, Ruby).
- ✔ **Systemnahe Sprachen (Low-Level)**: Näher an der Hardware (C, Assembly).
- ✔ **Objektorientierte Sprachen**: Arbeiten mit Klassen/Objekten (Java, C++, Python).
- ✔ **Prozedurale/imperative Sprachen**: Fokus auf Ablauf in Funktionen/Prozeduren (C, Pascal).

- ✔ **Skriptsprachen**: Werden erst zur Laufzeit in Maschinenbefehle umgesetzt (»interpretiert«), gut für Automatisierung (Python, Perl, Bash, JavaScript).
- ✔ **Auszeichnungssprachen**: Strukturieren Inhalte, keine Logik (HTML, XML).
- ✔ **Abfragesprachen**: Für Datenbanken, zum Beispiel SQL.

HTML ist keine Programmiersprache, sondern eine Auszeichnungssprache. Sie definiert Struktur und Inhalt einer Webseite. Berechnungen oder Wenn-dann-Verzweigungen bieten erst Programmiersprachen wie JavaScript.

Für unser Spiel nutzen wir HTML und CSS als Gerüst; die Spiellogik programmieren wir in JavaScript. JavaScript läuft im Browser und eignet sich sehr gut für interaktive Anwendungen. Es gehört zu den meistgenutzten Sprachen weltweit.

Eine Webseite nutzt typischerweise *HTML*, *CSS* und *JavaScript*:

- ✔ HTML bestimmt die Struktur der Seite,
- ✔ CSS bestimmt das Design und
- ✔ JavaScript kümmert sich um die Interaktivität.

Tag 6: Quiz – testen Sie Ihr Wissen

Frage 1: Im folgenden Code hat sich der Fehlerteufel eingeschlichen. Finden Sie den Fehler! Was ist falsch und wozu führt es?

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

Frage 2: HTML ist die Grundlage jeder Webseite. Zu welcher Sprachfamilie gehört HTML?

1. Programmiersprache
2. Auszeichnungssprache
3. Skriptsprache
4. Datenbanksprache

Frage 3: Nach dem Neuladen einer Webseite sind Ihre Änderungen nicht sichtbar. Woran kann das liegen?

1. Der Browser hat die alte Version der Datei im Cache gespeichert.
2. Die Änderungen waren nur temporär in den Entwicklertools und wurden nicht dauerhaft gespeichert.
3. Der Webserver ist nicht erreichbar.
4. Erst wenn man auf Speichern klickt, werden die Änderungen übernommen.

Frage 4: Sie haben in Ihrem Programm eine Variable namens `x`. Wie geben Sie den Wert dieser Variablen in der Konsole aus?

1. `echo x;`
2. `print(x);`
3. `console.log(x);`
4. `console.print(x);`

Frage 5: In einem Programm stolpern Sie über die folgende Codezeile. Was bewirkt dieser Befehl?

```
prompt("Wie heißt du?");}
```

1. Es wird eine Eingabeaufforderung angezeigt.
2. Der Code wirft einen Fehler, weil der Befehl nicht existiert.
3. Es wird eine Nachricht in der Konsole angezeigt.
4. Es passiert nichts, weil die Eingabe nicht in einer Variablen gespeichert wird.