Kapitel 1: Sie arbeiten bereits mit Markdown – ohne es zu wissen

Wenn Sie in den letzten Monaten eine KI um Hilfe bei einem Text gebeten haben, ist Ihnen vermutlich aufgefallen, dass die Antworten erstaunlich gut formatiert erscheinen. Überschriften sind deutlich größer und fetter als der Fließtext. Listen sind sauber eingerückt. Tabellen wirken ordentlich strukturiert. Code-Beispiele heben sich optisch ab. Das alles geschieht nicht zufällig.

Was Sie im Chat-Fenster sehen, ist das Ergebnis eines Zwischenschritts. Die KI schreibt ihren Text nicht direkt in dem Format, in dem Sie ihn sehen. Sie erzeugt zunächst einen strukturierten Text in einem Format namens Markdown. Erst die Chat-Oberfläche, zum Beispiel in Ihrem Browser, nimmt diesen Text und wandelt ihn in die ansprechende Darstellung um, die Sie auf dem Bildschirm sehen. Dieser Prozess heißt “Rendering” – aus Struktur wird formatierte Darstellung.

Sie können das leicht überprüfen. Bitten Sie eine KI um einen kurzen Text mit einer Überschrift, einer Liste und einer kleinen Tabelle. Die Antwort wird vermutlich professionell aussehen: Die Überschrift ist größer, die Liste hat Aufzählungszeichen, die Tabelle zeigt klare Spalten und Zeilen. Nun bitten Sie um denselben Text, aber mit der expliziten Anweisung: “Zeige mir den Markdown-Code dafür, nicht die gerenderte Version.”

Was Sie jetzt sehen, sieht auf den ersten Blick nüchterner aus:

```
# Einkaufsliste

Heute noch zu besorgen:
- Milch
- Brot
- Käse

| Artikel | Menge | Preis |
|---------|-------|-------|
| Milch   | 1 L   | 1,29€ |
| Brot    | 500g  | 2,50€ |
```

Das ist Markdown. Ein einfaches Textformat, bei dem Struktur durch Zeichen markiert wird. Das # kennzeichnet eine Überschrift. Der Bindestrich - am Zeilenanfang markiert einen Listenpunkt. Die senkrechten Striche | und die Zeile mit Bindestrichen strukturieren eine Tabelle. Konvertiert sieht das dann so aus:

Einkaufsliste

Heute noch zu besorgen: - Milch - Brot
- Käse

Artikel	Menge	Preis
Milch	1 L	1,29€
Brot	500g	2,50€

Der entscheidende Punkt: Die KI hat diesen Code von Anfang an erzeugt. Was Sie normalerweise im Chat sehen, ist nur die grafisch aufbereitete Form davon. Ihr Browser liest den Markdown-Code, interpretiert die Strukturzeichen und zeigt Ihnen das Ergebnis: eine größere Überschrift, eingerückte Listenpunkte, eine formatierte Tabelle.

Warum Copy-Paste trotzdem nicht funktioniert

Wenn Sie jetzt Text aus dem Chat-Fenster kopieren und in Word oder eine andere Textverarbeitung einfügen, greifen Sie nicht auf den ursprünglichen Markdown-Code zu. Sie kopieren die bereits gerenderte, vom Browser aufbereitete Version. Das ist etwa so, als würden Sie ein gedrucktes Buch abfotografieren statt die Originaldatei zu verwenden, aus der es gedruckt wurde.

Word versucht nun zu verstehen, was es da bekommt. Es sieht formatierten Text – aber ohne die strukturelle Information, die im Markdown-Code enthalten war. Eine Überschrift ist für Word nur “etwas größerer, fetter Text”. Eine Liste ist “eine Ansammlung von Zeilen, die zufällig mit einem Strich beginnen”. Eine Tabelle ist im schlimmsten Fall “Text, der durch viele Leerzeichen ausgerichtet wurde”.

Das Ergebnis kennen Sie: Word macht, was es kann, aber das Ergebnis entspricht selten dem, was Sie brauchen. Überschriften müssen nachträglich als Überschriften formatiert werden. Listen verlieren ihre Struktur. Tabellen müssen komplett neu aufgebaut werden.

Man könnte nun versuchen, direkt an den Markdown-Code zu kommen, den die KI ursprünglich erzeugt hat. Manche Chat-Interfaces bieten “Copy”-Buttons an. Aber diese kopieren in der Regel ebenfalls die gerenderte Version, nicht den Quelltext. Man könnte in den Entwicklertools des Browsers graben oder Browser-Erweiterungen verwenden. Das ist möglich, aber umständlich und wenig praktikabel für den alltäglichen Gebrauch.

Was macht Markdown besonders?

Markdown ist keine Erfindung für KI-Systeme. Es existiert seit 2004 und wurde ursprünglich entwickelt, um Texte für Webseiten zu schreiben, ohne HTML-Code von Hand tippen zu müssen. Die Grundidee war: Text soll als Klartext lesbar bleiben, auch wenn man strukturelle Informationen hinzufügt.

Vergleichen Sie das mit HTML, das Sie vielleicht vom Blick in den Quellcode einer Webseite kennen. Eine einfache Überschrift sieht dort so aus:

```
<h1>Überschrift</h1>
```

Eine Liste wird zu:

```
<ul>
  <li>Erster Punkt</li>
  <li>Zweiter Punkt</li>
  <li>Dritter Punkt</li>
</ul>
```

Das ist präzise und eindeutig, aber für den Menschen mühsam zu lesen und zu schreiben. Markdown reduziert das auf das Wesentliche:

```
# Überschrift
```

```
- Erster Punkt
- Zweiter Punkt
- Dritter Punkt
```

Der Text bleibt lesbar. Die Struktur ist trotzdem klar definiert. Genau diese Kombination macht Markdown für KI-Systeme ideal. Sie können es zuverlässig erzeugen, weil die Regeln eindeutig sind. Und Menschen können es ohne Vorwissen zumindest lesen, auch wenn das Schreiben anfangs ungewohnt erscheint.

Die gute Nachricht: Sie müssen Markdown nicht selbst schreiben lernen. Die KI übernimmt das für Sie. Was Sie verstehen sollten, ist die Grundlogik des Formats – und wie Sie die KI so instruieren, dass sie Ihnen genau das liefert, was Sie brauchen. Darum geht es in den nächsten Kapiteln.

Doch zunächst lohnt es sich, die grundlegenden Elemente von Markdown kennenzulernen. Nicht um sie selbst zu schreiben, sondern um zu verstehen, was die KI Ihnen liefert – und um beurteilen zu können, ob das Ergebnis korrekt ist. Sie müssen die Syntax nicht detailliert lernen. Sie können den folgenden Abschnitt auch zunächst einmal nur überfliegen und denn bei Bedarf darauf zurückkommen.

Überschriften

Überschriften werden durch Rautezeichen am Zeilenanfang markiert. Ein Rautezeichen steht für eine Überschrift erster Ebene, zwei für die zweite Ebene, und so weiter bis zur sechsten Ebene.

```
# Überschrift Ebene 1
## Überschrift Ebene 2
### Überschrift Ebene 3
```

Das entspricht in Word den Formatvorlagen “Überschrift 1”, “Überschrift 2” und “Überschrift 3”. Der Unterschied: In Markdown ist die Hierarchie im Quelltext sofort erkennbar. Man sieht auf einen Blick, welche Ebene gemeint ist.

Wichtig ist der Abstand zwischen Rautezeichen und Text. Es sollte immer ein Leerzeichen dazwischen stehen. `# Überschrift` ist korrekt, `#Überschrift` kann zu Problemen führen. Die meisten KI-Systeme setzen diesen Abstand automatisch, aber es lohnt sich, bei Kontrolle darauf zu achten.

Textauszeichnungen

Fettdruck wird durch doppelte Sternchen oder doppelte Unterstriche erzeugt:

```
**fetter Text** oder __fetter Text__
```

Kursivdruck durch einfache Sternchen oder einfache Unterstriche:

```
*kursiver Text* oder _kursiver Text_
```

Beides lässt sich kombinieren:

```
***fett und kursiv***
```

In der Praxis verwenden die meisten KI-Systeme Sternchen, weil diese visuell klarer sind. Das ist konsistent und führt selten zu Fehlern. Wenn Sie allerdings Texte manuell nachbearbeiten, sollten Sie bei einer Variante bleiben – entweder Sternchen oder Unterstriche, nicht beides gemischt.

Listen

Ungeordnete Listen beginnen mit einem Bindestrich, Sternchen oder Pluszeichen:

```
- Erster Punkt
- Zweiter Punkt
- Dritter Punkt
```

Geordnete Listen verwenden Zahlen mit einem Punkt:

```
1. Erster Punkt
2. Zweiter Punkt
3. Dritter Punkt
```

Bei geordneten Listen ist ein Detail interessant: Die tatsächlichen Zahlen im Quelltext spielen keine Rolle. Man könnte auch schreiben:

```
1. Erster Punkt
2. Zweiter Punkt
3. Dritter Punkt
```

Der Konverter nummeriert automatisch korrekt durch. Das ist praktisch, wenn man nachträglich Punkte einfügt oder die Reihenfolge ändert. Man muss nicht alle Zahlen neu durchzählen.

Listen können verschachtelt werden. Dafür wird eingerückt, in der Regel mit zwei oder vier Leerzeichen:

```
- Hauptpunkt
  - Unterpunkt
  - Noch ein Unterpunkt
- Nächster Hauptpunkt
```

Hier wird es etwas fehleranfälliger. Nicht alle KI-Systeme rücken konsistent ein, manche verwenden zwei Leerzeichen, andere vier, wieder andere Tabs. Für die meisten Konverter spielt das keine Rolle, solange die Einrückung einheitlich bleibt. Wenn Sie allerdings Probleme bei der Konvertierung bemerken, lohnt sich ein Blick auf die Einrückungstiefe.

Tabellen

Tabellen sind eines der praktischsten Elemente in Markdown – und gleichzeitig eines der anfälligsten für Formatierungsfehler. Die Grundstruktur ist einfach:

```
| Spalte 1 | Spalte 2 | Spalte 3 |
|----------|----------|----------|
| Wert 1   | Wert 2   | Wert 3   |
| Wert 4   | Wert 5   | Wert 6   |
```

Die erste Zeile enthält die Spaltenüberschriften. Die zweite Zeile besteht aus Bindestrichen und trennt Kopf von Körper. Ab der dritten Zeile folgen die Datenzeilen.

Die senkrechten Striche müssen nicht exakt untereinander stehen. Folgendes funktioniert genauso:

```
| Spalte 1 | Spalte 2 | Spalte 3 |
|---|---|---|
| Wert 1 | Wert 2 | Wert 3 |
```

Die Ausrichtung der Striche dient nur der Lesbarkeit im Quelltext. Für die Konvertierung spielt sie keine Rolle. KI-Systeme erzeugen Tabellen manchmal akkurat ausgerichtet, manchmal kompakt. Beides ist korrekt.

Was allerdings wichtig ist: Die Anzahl der Spalten muss in allen Zeilen übereinstimmen. Wenn die Kopfzeile drei Spalten definiert, muss jede Datenzeile ebenfalls drei Spalten haben. Fehlt eine, kann die Tabelle beim Konvertieren zerfallen.

Die Trennzeile kann auch Ausrichtung definieren. Doppelpunkte steuern, ob eine Spalte linksbündig, rechtsbündig oder zentriert erscheint:

```
| Links | Zentriert | Rechts |
|:------|:----------:|-------:|
| Text  | Text       | Text   |
```

`:---` bedeutet linksbündig (Standard), `:---:` zentriert, `---:` rechtsbündig. Das funktioniert bei der Konvertierung zu HTML oder PDF zuverlässig. Bei Word-Formaten hängt es vom verwendeten Konverter ab, ob die Ausrichtung übernommen wird.

Links	Zentriert	Rechts
Text	Text	Text

Tabellen sind ein gutes Beispiel dafür, warum präzises Prompting wichtig ist. Wenn Sie eine KI bitten, eine Tabelle zu erstellen, bekommt sie das in der Regel hin. Aber wenn Sie nicht spezifizieren, wie viele Spalten, welche Überschriften und welche Ausrichtung gewünscht sind, kann das Ergebnis überraschend ausfallen. Dazu später mehr im Kapitel über effektives Prompting.

Mathematische Formeln

Hier wird Markdown um eine Konvention aus der LaTeX-Welt erweitert. LaTeX ist ein Textsatzsystem, das vor allem in der Wissenschaft für mathematische Dokumente verwendet wird. Markdown übernimmt dessen Syntax für Formeln.

Inline-Formeln – also Formeln im Fließtext – werden in einfache Dollarzeichen eingeschlossen:

```
Die Fläche eines Kreises beträgt $A = \pi r^2$.
```

Die Fläche eines Kreises beträgt $A = \pi r^2$.

Display-Formeln, die als eigener Block gesetzt werden, verwenden doppelte Dollarzeichen:

```
$$
E = mc^2
$$
```

Die Syntax innerhalb der Dollarzeichen folgt LaTeX-Regeln. Griechische Buchstaben beginnen mit einem Backslash: `\pi`, `\alpha`, `\beta`. Brüche werden mit `\frac{Zähler}{Nenner}` geschrieben. Hochstellungen mit `^`, Tiefstellungen mit `_`.

Ein etwas komplexeres Beispiel:

```
$$
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
$$
```

Das ist die Mitternachtsformel für quadratische Gleichungen. Im Quelltext sieht das auf den ersten Blick kryptisch aus. Nach der Konvertierung erscheint sie als sauber gesetzte mathematische Formel.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Auf die gleiche Weise können Integrale, Vektoren, Matrizen und andere mathematische Elemente erstellt werden. Vielleicht haben Sie Fragmente davon mal auftreten sehen, wenn Sie Ausgabetext mit copy-paste übertragen wollten.

Für Sie als Nutzer ist wichtig: Sie müssen diese Syntax nicht beherrschen. Die KI kann das. Aber Sie sollten wissen, dass Formeln in Dollarzeichen stehen und dass dort eine andere Syntax gilt als im normalen Text. Wenn Sie eine Formel kontrollieren, prüfen Sie vor allem, ob die Dollarzeichen korrekt gesetzt sind und ob die Struktur plausibel aussieht.

Code

Markdown unterscheidet zwischen Inline-Code und Code-Blöcken. Inline-Code steht in einfachen Backticks:

```
Die Funktion `print()` gibt Text aus.
```

Code-Blöcke verwenden dreifache Backticks:

````
```
def beispiel():
    print("Hallo Welt")
```
````

Optional kann man die Programmiersprache angeben:

````
```python
def beispiel():
    print("Hallo Welt")
```
````

Das ermöglicht Syntax-Highlighting bei der Konvertierung zu HTML oder PDF. Für Word-Formate wird Code in der Regel als Festbreitenschrift wie Courier formatiert.

Code-Blöcke sind praktisch, wenn Sie Beispiele, Konfigurationsdateien oder ähnliches in einem Dokument darstellen wollen. Die KI erzeugt sie zuverlässig, vor allem wenn man explizit darum bittet.

Da die Markup-Quelltextausgabe selbst Code ist, kann darin enthaltener Code zu Verwerfungen führen. Wir gehen darauf in einem eigenen Abschnitt genauer ein.

Bilder

Bilder werden in Markdown mit einer speziellen Syntax eingebunden:

```
![Alternativtext](pfad/zum/bild.png)
```

Der Alternativtext in eckigen Klammern erscheint, wenn das Bild nicht geladen werden kann. Der Pfad in runden Klammern verweist auf die Bilddatei.

In der Praxis verwenden Sie hier meist Platzhalter, wenn Sie mit KI arbeiten. Die KI kann keine Bilder erzeugen oder einbinden – dazu fehlt ihr der Zugriff auf Ihr Dateisystem. Aber sie kann Platzhalter setzen:

```
![Diagramm zur Umsatzentwicklung](diagramm_umsatz.png)
```

Nach der Konvertierung ersetzen Sie den Platzhalter durch das tatsächliche Bild, oder Sie erstellen das Bild und speichern es unter dem angegebenen Namen im richtigen Verzeichnis.

Links

Links funktionieren ähnlich wie Bilder, nur ohne das führende Ausrufezeichen:

```
[Linktext](https://beispiel.de)
```

Das ist relevant, wenn Sie Dokumente mit Verweisen erstellen – etwa Handbücher mit Links zu weiterführenden Ressourcen.

Die einfachere Lösung: Bewusst Markdown anfordern

Es gibt einen direkteren Weg, der zudem zuverlässiger funktioniert. Statt zu versuchen, nachträglich an den Markdown-Code zu kommen, fordert man ihn explizit an.

Der Prompt ist einfach:

> “Erstelle einen Bericht über [Thema]. Gib mir das Ergebnis als Markdown-Code, nicht gerendert.”

Oder noch präziser:

> “Schreibe [Text]. Formatiere das Ergebnis in Markdown und zeige mir den Quellcode, den ich direkt kopieren kann.”

Die KI liefert dann den reinen Markdown-Text. Nicht die aufbereitete Browser-Darstellung, sondern den tatsächlichen Code mit allen Strukturzeichen. Dieser Text lässt sich problemlos kopieren – es ist schlichter Klartext, keine komplexe Formatierung, die beim Übertragen verloren gehen könnte.

Und genau dieser Text ist es, den Sie anschließend weiterverarbeiten können. In eine Datei speichern. Mit einem Konverter in Word, LibreOffice oder HTML umwandeln. Oder einfach mit einem Texteditor öffnen und bei Bedarf anpassen, bevor Sie ihn konvertieren.

Warum diese Elemente ausreichen

Beim ersten Kontakt mit Markdown fragen sich viele: Wo sind die Schriftarten? Die Farben? Die Seitenumbrüche? Die Fußnoten? Die Textrahmen?

Die Antwort: Markdown ist bewusst minimalistisch. Es beschreibt Struktur, nicht Erscheinung. Schriftart, Schriftgröße, Farben – all das wird später durch Formatvorlagen oder Stylesheets festgelegt. Nicht im Quelltext.

Das mag zunächst wie eine Einschränkung wirken. Tatsächlich ist es ein Vorteil. Wenn Sie einen Text in Markdown schreiben (oder schreiben lassen), konzentrieren Sie sich auf den Inhalt

und die Struktur. Ist das eine Überschrift erster oder zweiter Ebene? Gehört dieser Absatz in eine Liste? Ist das eine Tabelle oder eher Fließtext?

Die visuelle Gestaltung kommt später. Und sie lässt sich ändern, ohne den Inhalt anzufassen. Wenn Sie heute ein Word-Dokument mit 14-Punkt-Arial erstellen und morgen feststellen, dass es in 12-Punkt-Calibri sein sollte, müssen Sie das Dokument neu formatieren. Wenn Sie aus Markdown konvertieren, ändern Sie einfach die Formatvorlage und konvertieren erneut. Der Inhalt bleibt unberührt.

Das ist auch der Grund, warum Markdown für KI-generierte Texte so gut funktioniert. Die KI muss keine Entscheidungen über Schriftarten oder Abstände treffen. Sie strukturiert den Text. Den Rest erledigen Sie – oder Ihr Konverter.

Ein erster Versuch

Bevor es im nächsten Kapitel darum geht, wie Sie die KI instruieren, können Sie bereits jetzt einen ersten Versuch unternehmen.

Öffnen Sie Ihr bevorzugtes KI-System und geben Sie folgenden Prompt ein:

> “Erstelle eine kurze Übersicht (ca. 200 Wörter) über die Vorteile erneuerbarer Energien. Gliedere den Text mit einer Hauptüberschrift, zwei Unterüberschriften und jeweils einer kurzen Liste von Vorteilen. Gib mir das Ergebnis als Markdown-Code in einem Codefeld aus.”

Die KI wird Ihnen einen Text liefern, der ungefähr so aussieht:

```
# Vorteile erneuerbarer Energien

## Ökologische Vorteile

- Keine CO₂-Emissionen während der Stromerzeugung
- Schonung fossiler Ressourcen
- Verringerung der Luftverschmutzung

## Ökonomische Vorteile

- Sinkende Kosten durch technologische Entwicklung
- Unabhängigkeit von Rohstoffimporten
- Schaffung neuer Arbeitsplätze
```

Kopieren Sie diesen Text in einen einfachen Texteditor – etwa Notepad unter Windows oder TextEdit unter macOS. Speichern Sie die Datei mit der Endung `.md`, also zum Beispiel `erneuerbare_energien.md`.

Sie haben gerade Ihr erstes Markdown-Dokument erstellt. Es ist eine schlichte Textdatei. Wenn Sie sie in einem Texteditor öffnen, sehen Sie den Quellcode mit den Strukturzeichen. Das ist normal und gewollt.

Wenn Sie selbst Markdown schreiben oder bearbeiten wollen, gibt es dafür spezielle Editoren. So ist zum Beispiel auf https://www.zettlr.com ein spezieller Markdowneditor für die wichtigsten Betriebssysteme kostenlos verfügbar. Es gibt viele andere, die meisten beinhalten zusätzliche Möglichkeiten. Für Ihren eigentlichen Workflow benötigen Sie sie aber nicht.

Im übernächsten Kapitel werden Sie lernen, wie Sie dieses Dokument in ein formatiertes Word-, LibreOffice- oder HTML-Dokument verwandeln. Vorher aber lohnt es sich, tiefer in die Kunst des Promptings einzusteigen. Denn die Qualität des Markdown-Codes hängt maßgeblich davon ab, wie präzise Sie der KI sagen, was Sie brauchen.