PRAXIS-GUIDE

Präsentationen mit KI

Warum du aufhören solltest, PowerPoint zu verlangen, und was stattdessen funktioniert

Maik Schwede

maikschwede.de

Workflow, Prompt-Bausteine und Checkliste für Decks, die aus HTML entstehen und als PDF beim Empfänger landen.

Vorwort

Du kennst die Situation. Übermorgen ist der Termin, die Unterlagen fehlen, und du machst das, was alle machen: Du öffnest ein KI-Tool und tippst „Bau mir eine Präsentation über X“.

Was rauskommt, ist entweder langweilig oder kaputt. Meistens beides.

Lange dachte ich, ich wäre der Einzige, der damit kämpft. In vielen Gesprächen habe ich dann gemerkt, dass es fast allen so geht. Es redet nur keiner drüber, weil es sich anfühlt, als hätte man es selbst nicht richtig hinbekommen.

Über dieses Thema wird gern geschrieben, es sei der heilige Gral, an dem sich alle die Zähne ausbeißen. Ich drehe den Satz um. Gelöst ist es längst, nur nicht an der Stelle, an der die meisten suchen. Nicht im nächsten Präsentations-Tool, das man abonnieren kann, sondern eine Ebene tiefer: in der Frage, in welchem Format eine Maschine überhaupt vernünftig arbeiten kann.

Denn das ist der Kern. Wir verlangen von der KI ein Dateiformat, das für menschliche Handarbeit gebaut wurde, und wundern uns dann über das Ergebnis. Falscher Prompt, falsches Tool, zu wenig Geduld, all das ist es nicht. Es ist ein technisches Problem, und deshalb hat es auch eine technische Lösung.

Der Ausweg klingt erstmal nach einem Umweg: Du lässt die KI nicht PowerPoint bauen, sondern Code. Genauer gesagt HTML und CSS. Am Ende kommt trotzdem ein PDF raus, das du verschicken kannst. Nur ist der Weg dahin ein völlig anderer, und die Qualität auch.

Bei mir läuft dieser Ablauf im Tagesgeschäft, und ganz ehrlich, er ist der einfachste Teil davon. Nach demselben Prinzip entstehen hier Videos, komplette Websites, vertonte Artikel, Systeme, die weiterarbeiten, während ich woanders bin. Gemessen daran sind ein paar Folien ein Kindergeburtstag.

Genau deshalb fange ich damit an. Folien sind der Einstieg, an dem du das Prinzip begreifst. Wer es einmal verstanden hat, wendet es auf alles andere an.

Dieses E-Book ist die Anleitung dazu. Kein Theorie-Vortrag, sondern der Ablauf, den du morgen früh nachbauen kannst. Mit Prompts zum Kopieren, mit den Stellen, an denen es typischerweise schiefgeht, und mit einer Checkliste am Ende.

Eine Sache vorweg: Die KI übernimmt hier die Mitte. Nicht den Anfang, nicht das Ende. Was du sagen willst, wem du es sagst und ob das Ergebnis gut ist, bleibt deine Arbeit. Wer das an die Maschine abgibt, bekommt genau den austauschbaren Einheitsbrei zurück, über den sich alle beschweren.

Ich hoffe, es hilft dir weiter.

Dein Maik

Kapitel 1: Warum die KI an deiner PowerPoint scheitert

Ich fange mit der Technik an, weil das Verständnis dafür den Rest der Anleitung erklärt.

Eine .pptx-Datei sieht aus wie ein Dokument. Ist sie aber nicht. Sie ist ein Paket. Wenn du eine .pptx entpackst, findest du keinen zusammenhängenden Text, sondern einen Haufen einzelner XML-Dateien, die sich gegenseitig referenzieren.

Microsoft beschreibt das in der eigenen Entwicklerdokumentation so: Ein PresentationML-Dokument wird nicht als ein großer Block in einer einzigen Datei gespeichert. Stattdessen liegen die Funktionsgruppen in getrennten Teilen, und für jede einzelne Folie entsteht eine eigene XML-Datei.

Dazu kommen Slide Master, Slide Layout, Theme, Notes Master, Handout Master, dazu Medien, dazu die Beziehungsdateien, die festlegen, welcher Teil auf welchen anderen zeigt. Damit eine Präsentation überhaupt gültig ist, müssen mindestens presentation, sldMaster, sldLayout, sld und theme sauber zusammenspielen.

Und jetzt stell dir vor, was die KI leisten soll.

Sie soll dein Thema verstehen. Sie soll daraus eine Dramaturgie entwickeln. Sie soll die Folien gestalten, also Hierarchie, Weißraum, Typografie, Rhythmus, Reduktion. Und sie soll das Ergebnis in ein verschachteltes Dateiformat schreiben, das für die manuelle Bearbeitung durch Menschen gebaut wurde. Alles gleichzeitig, ohne dabei sehen zu können, was auf der Folie passiert.

Das ist, als würdest du einen Architekten bitten, ein Haus zu entwerfen, und ihm dabei die Augen verbinden und nur einen Bleistift geben, mit dem er Koordinaten diktieren darf.

Deshalb landest du in einem von zwei Ergebnissen. Entweder das Tool spielt auf Nummer sicher, dann bekommst du formal korrekte Folien, die aussehen wie aus dem Baukasten. Oder das Tool wird ambitionierter, dann brechen Abstände, Schriftgrößen und Bildlogik auseinander, und du reparierst zwei Stunden lang, was du in einer Stunde selbst gebaut hättest.

Wann PowerPoint trotzdem das richtige Endformat ist

Ich will hier nicht gegen PowerPoint predigen. In vielen Firmen ist die .pptx das Arbeitsformat, weil vier Leute daran weiterarbeiten und Folien untereinander tauschen. Dann brauchst du .pptx. Punkt.

Die Frage ist nur, ob das bei dir wirklich der Fall ist. In sehr vielen Situationen soll überhaupt niemand die Datei bearbeiten. Du zeigst sie, du schickst sie hinterher, fertig. Für diesen Fall ist die .pptx nur noch Gewohnheit, kein Bedarf.

Und wenn es kein Bedarf ist, kannst du der KI ein Format geben, mit dem sie deutlich besser arbeitet.

Kapitel 2: HTML als Arbeitsformat

HTML ist die Struktursprache des Webs. Sie beschreibt, welche Elemente es gibt und wie sie zueinander stehen. Überschrift, Absatz, Bild, Liste, Abschnitt. CSS ist die Gestaltungssprache dazu. Sie legt fest, wie diese Elemente aussehen. Farbe, Schrift, Abstände, Raster, Größen, Position, Druckverhalten.

Der entscheidende Unterschied zu PowerPoint ist nicht technische Eleganz. Er ist praktischer Natur.

Bei einer .pptx steuerst du das Ergebnis. Du sagst „mach es schön“ und hoffst.

Bei HTML und CSS gibst du ein Regelwerk vor. Farbwerte, Schriftgrößen-Skala, Abstandsraster, Komponenten, Druckregeln. Die KI wendet dieses Regelwerk an, und zwar konsequent auf jede Folie, weil es als Text im Code steht und nicht als Gefühl im Prompt.

Das ist der eigentliche Hebel. Nicht das Format, sondern die Tatsache, dass Geschmack plötzlich schreibbar wird.

Ich sehe darin ein Muster, das weit über Präsentationen hinausgeht: Alles, was du bei der Arbeit mit KI im Kopf behältst, ist verloren. Alles, was du aufschreibst, ist ein Werkzeug. Designlogik, Qualitätsmaßstäbe, Storystruktur, deine Marotten. Solange das im Kopf bleibt, korrigierst du bei jedem Durchlauf von vorne. Sobald es geschrieben steht, korrigierst du einmal und profitierst hundertmal davon.

Drei praktische Vorteile kommen dazu.

Du siehst sofort, was passiert. HTML rendert im Browser. Die meisten KI-Oberflächen zeigen dir das Ergebnis direkt in einer Vorschau. Bei Claude heißen diese Vorschauen Artifacts, und einseitige HTML-Seiten sind ausdrücklich einer der unterstützten Typen. Voraussetzung ist, dass in den Einstellungen die Code-Ausführung freigeschaltet ist.

Du kannst punktuell korrigieren. „Die Zahlen auf Folie 7 sind zu klein“ führt zu einer Änderung an einer CSS-Zeile, nicht zu einem kompletten Neubau des Decks.

Du kannst das Regelwerk wiederverwenden. Beim nächsten Deck startest du nicht bei null, sondern mit deinem Designsystem.

Kapitel 3: Der Workflow

Vier Schritte. Der Mensch steht am Anfang und am Ende, die KI arbeitet in der Mitte.

Schritt 1: Die Designreferenz auseinandernehmen

Fang nicht mit einer leeren Folie an. Fang mit einer visuellen Referenz an.

Das kann fast alles sein: ein Deck, das dir gefällt, deine eigene Website, eine Landingpage, ein PDF-Report, ein Brand Guide, ein Screenshot. Wichtig ist nur, dass die Wirkung stimmt, die du treffen willst.

Die KI soll daraus nicht kopieren. Sie soll die Logik dahinter ableiten und in eine Guideline schreiben, die du danach jedes Mal wiederverwendest. Welche Farben, in welchem Verhältnis, mit welchen Regeln. Welche Schriftgrößen, in welchen Sprüngen. Wie viel Weißraum. Wo sitzt der Footer. Wie sieht eine Karte aus, wie ein Badge, wie eine Tabelle.

Das Ergebnis speicherst du als Markdown-Datei ab. Diese Datei ist ab jetzt dein Designsystem.

Ein Tipp aus der Praxis: Lass dir am Ende der Guideline sogenannte Design-Tokens ausgeben. Das sind CSS-Variablen für Farben, Schriftgrößen, Abstände, Radien und Schatten. Die kann die KI im nächsten Schritt eins zu eins in den Code übernehmen, und du hast eine zentrale Stelle, an der du später alles änderst.

Schritt 2: Erst denken, dann bauen

Hier passiert der häufigste Fehler, und er kostet die meiste Zeit.

Die meisten Leute lassen die KI direkt Folien bauen, obwohl die Präsentation noch gar nicht gedacht ist. Dann sitzt du vor 18 hübschen Folien und merkst, dass die Argumentation nicht trägt. Jetzt kannst du entweder alles wegwerfen oder dich in die Aussage einer Präsentation reinfummeln, die schon fertig aussieht. Beides schlecht.

Trenne deshalb sauber. In diesem Schritt entsteht kein Design, sondern Architektur.

Was du brauchst:

- die Kernbotschaft in einem Satz, also das, was hängenbleiben soll, wenn alles andere vergessen ist
- eine Dramaturgie in fünf bis sieben Stationen
- eine Tabelle aller Folien mit Nummer, Titel, Kernaussage, Visual-Idee und Folientyp
- pro Folie die Funktion im roten Faden, also warum diese Folie überhaupt existiert
- ein Briefing für den Bau-Schritt

Wenn eine Folie keine Funktion im roten Faden hat, fliegt sie raus. Diese eine Regel spart dir bei jedem Deck drei bis fünf Folien.

Schritt 3: Das Deck als Code bauen lassen

Jetzt wird gebaut. Die KI bekommt drei Eingaben:

1. die Design-Guideline aus Schritt 1, inklusive der CSS-Tokens
2. die Folienarchitektur aus Schritt 2
3. das Briefing, das am Ende von Schritt 2 entstanden ist

Und dann kommt der Teil, der über die Qualität entscheidet: Die KI darf nicht sofort losschreiben.

Lass sie erst einen Plan machen. Welche Folientypen werden gebraucht. Welche Komponenten. Welche Folie nutzt welchen Typ. Wo droht Ärger, weil zu viel Text auf eine Folie soll oder weil ein Element beim PDF-Export brechen wird.

Dieser Vorab-Plan dauert dreißig Sekunden und verhindert das, was sonst passiert: Die KI baut Folie 1 bis 4 schön, erfindet ab Folie 5 neue Abstände, und ab Folie 9 sieht das Deck aus wie von zwei verschiedenen Leuten.

Erst nach dem Plan entsteht Code, und zwar in dieser Reihenfolge: Design-Tokens als CSS-Variablen, Grundlayout der Folie, Komponenten, Folientypen, Druckregeln. Die einzelnen Folien kommen ganz zum Schluss. Wenn das Fundament steht, sind die Folien nur noch das Ausfüllen von Vorlagen.

Schritt 4: Iterieren und das Gelernte zurückschreiben

Du schaust dir das Ergebnis an und gibst Feedback. Konkret, nicht als Gefühl. „Die Überschrift auf Folie 3 hat zu wenig Luft nach unten“ bringt dich weiter als „wirkt gedrängt“.

Nach zwei, drei Runden kommt der Schritt, den fast alle auslassen: Lass die KI die Korrekturen als Regeln in deine Guideline zurückschreiben.

Wenn du bei jedem Deck sagen musst, dass Zahlen größer gesetzt werden sollen, dann gehört das nicht in den nächsten Prompt, sondern als Regel in die Datei. Beim zehnten Deck sagst du nichts mehr, weil alles drinsteht.

Das ist der Punkt, an dem der Aufwand kippt. Das erste Deck kostet dich mehr Zeit als der alte Weg. Das dritte kostet weniger. Ab dem fünften ist es nicht mehr vergleichbar.

Kapitel 4: Prompt-Bausteine zum Kopieren

Vier Bausteine, die du direkt übernehmen kannst. Ersetze die Platzhalter in eckigen Klammern.

Baustein 1: Designreferenz zerlegen

```
Analysiere die beigefügte Designreferenz und leite daraus eine
Design-Guideline für HTML-Präsentationen ab.

Wichtig: Nichts kopieren. Ich will die dahinterliegende Logik,
nicht das Original.

Regeln fuer deine Antwort:
- Jede Aussage als konkrete Regel mit Wert, nicht als Beschreibung.
  Also "Abstand zwischen Headline und Body: 24 px" statt
  "grosszuegiger Abstand".
- Wenn du etwas aus der Referenz nicht erkennen kannst, markiere es
  ausdruecklich als Annahme. Nicht raten.

Liefere als Markdown mit diesen Abschnitten:
1. Wirkung in einem Satz, dazu 3 bis 5 Merkmale, an denen man diesen
   Stil erkennt
2. Farbpalette mit Hex-Werten, Einsatzregeln und ausdruecklichen No-Gos
3. Typografie-Skala: H1, H2, Body, Label, Zahlen. Groesse,
   Schriftschnitt, Zeilenhoehe, Laufweite
4. Layoutsystem: Seitenverhaeltnis, Raster, Aussenabstaende,
   Weissraumregeln, Kopf- und Fusszeile
5. Komponenten: Karte, Badge, Tabelle, Zitat, Kennzahl. Jeweils
   Aufbau und Regeln
6. Bildsprache: Icons, Fotos, Illustrationen, Diagramme
7. 6 bis 10 Folientypen mit Zweck, Layout und visueller Logik
8. 10 harte Regeln, die nie gebrochen werden duerfen
9. Qualitaetscheckliste zum Abhaken
10. CSS-Tokens als :root-Variablen fuer Farben, Typografie,
    Abstaende, Radien, Schatten, Linien
```

Baustein 2: Folienarchitektur

```
Du bist Praesentationsarchitekt. Wir bauen noch keine Folien,
wir bauen die Struktur.

Kontext:
- Thema: [THEMA]
- Zielgruppe: [WER SITZT IM RAUM, WAS WEISS DIE PERSON SCHON,
  WAS INTERESSIERT SIE]
- Ziel: [WAS SOLL NACH DER PRAESENTATION PASSIEREN]
- Gewuenschte Wirkung: [WIE SOLL SICH DIE ZIELGRUPPE FUEHLEN]
- Dauer: [MINUTEN]
- Vorhandene Inhalte: [ZAHLEN, TEXTE, QUELLEN, ANHAENGE]
- Design-Guideline: [DATEI ANHAENGEN]

Liefere:
1. Kernbotschaft in einem Satz. Das, was haengenbleibt, wenn alles
   andere vergessen ist.
2. Dramaturgie in 5 bis 7 Stationen mit jeweils einer Zeile, was
   diese Station beim Publikum ausloest.
3. Folientabelle mit den Spalten: Nummer, Titel, Kernaussage in einem
   Satz, Visual-Idee, Folientyp aus der Guideline, Funktion im roten
   Faden.
4. Ein HTML-Briefing fuer den naechsten Schritt: welche Folientypen
   und Komponenten gebraucht werden und wo es beim Layout eng wird.

Harte Regeln:
- Maximal [ANZAHL] Folien.
- Jede Folie braucht eine Funktion im roten Faden. Wenn du keine
  benennen kannst, streich die Folie und sag mir warum.
- Erfinde keine Zahlen. Wenn eine Aussage eine Zahl braucht, die ich
  nicht geliefert habe, schreib [ZAHL FEHLT] hin.
```

Baustein 3: HTML-Deck bauen

```
Bau aus den drei Anhaengen eine HTML-Praesentation:
Design-Guideline, Folienarchitektur, HTML-Briefing.

Schritt 1, bevor du Code schreibst:
Zeig mir einen Plan. Welche Folientypen brauchst du, welche
Komponenten, welche Folie nutzt welchen Typ, und wo siehst du
Risiken bei Textmenge, Layout oder PDF-Export. Warte auf mein OK.

Schritt 2, nach meinem OK, in dieser Reihenfolge:
1. CSS-Tokens aus der Guideline als :root-Variablen
2. Grundlayout einer Folie plus Druckregeln
3. Komponenten
4. Folientypen
5. Erst dann die einzelnen Folien

Technische Vorgaben:
- Eine einzige HTML-Datei, alles inline. Keine externen Skripte,
  keine externen Schriften, keine CDN-Verweise.
- Jede Folie ist ein eigenes Element mit fester Groesse im
  Verhaeltnis 16:9.
- Druck-CSS von Anfang an mitdenken:
  @page { size: 338.67mm 190.5mm; margin: 0; }
  Ein Seitenumbruch nach jeder Folie.
  print-color-adjust: exact, damit Flaechen erhalten bleiben.
- Kein Text darf ueber den Folienrand hinauslaufen. Wenn eine Folie
  zu voll wird, sag mir das, statt die Schrift kleiner zu machen.
- Keine Inhalte erfinden. Was nicht in der Architektur steht, kommt
  auch nicht auf die Folie.
```

Baustein 4: Feedback in die Guideline zurückschreiben

```
Wir haben jetzt [ANZAHL] Korrekturrunden hinter uns.

Geh die Aenderungen durch, die ich verlangt habe, und schreib sie
als dauerhafte Regeln in die Design-Guideline zurueck.

Fuer jede Regel:
- Wo genau sie in der Guideline hingehoert
- Der neue oder geaenderte Regeltext, konkret mit Wert
- Wenn sie eine bestehende Regel ersetzt: welche und warum

Nimm nur Korrekturen auf, die grundsaetzlich gelten. Einmalige
Aenderungen fuer dieses eine Deck bleiben draussen. Wenn du dir bei
einer Korrektur nicht sicher bist, ob sie grundsaetzlich oder
einmalig war, frag mich.
```

Kapitel 5: Vom HTML zur Datei, die du verschicken kannst

Das Deck steht im Browser. Jetzt brauchst du etwas, das du anhängen kannst.

Der schnellste Weg

Wenn dein KI-Tool Code ausführen und Dateien erzeugen darf, sag ihm einfach, es soll ein PDF bauen. Das funktioniert, kostet dich einen Satz, und für ein einzelnes Deck reicht es völlig aus.

Kann dein Tool das nicht, geht es über den Browser.

Der Weg über den Druckdialog

1. HTML-Datei herunterladen und im Browser öffnen
2. Drucken aufrufen (Strg+P beziehungsweise Cmd+P)
3. Ziel auf „Als PDF speichern“ stellen
4. Layout auf Querformat
5. **Ränder auf „Keine“**
6. Hintergrundgrafiken aktivieren
7. Skalierung auf Standard belassen
8. Speichern

Punkt 5 ist wichtig, und hier wird oft falsch beraten. Bei einem randlosen 16:9-Deck erzeugen Standardränder weiße Rahmen und eine Verkleinerung der Folie. Das Framework reveal.js, das seit Jahren HTML-Präsentationen macht, gibt in der eigenen Dokumentation genau diese Kombination vor: Ziel „Als PDF speichern“, Layout Querformat, Ränder „Keine“, Hintergrundgrafiken an. reveal.js weist außerdem darauf hin, dass der Export nur in Google Chrome bestätigt funktioniert. Wenn dein PDF also seltsam aussieht, probier es zuerst in Chrome, bevor du am Code suchst.

Punkt 6 hat einen technischen Grund. Browser dürfen beim Drucken standardmäßig sparen. Sie lassen Hintergrundbilder weg und passen Textfarben an, damit es auf weißem Papier gut lesbar ist. Für ein Dokument ist das sinnvoll, für ein Deck mit dunklen Flächen eine Katastrophe. Deshalb gehört im CSS `print-color-adjust: exact` an die richtigen Stellen, und im Dialog der Haken bei den Hintergrundgrafiken.

Die richtige Foliengröße

Wenn du willst, dass dein PDF exakt die Maße einer PowerPoint-Folie hat, nimm diese Werte. Microsoft gibt für das Widescreen-Format 13,333 mal 7,5 Zoll an, also 33,867 mal 19,05 Zentimeter.

Im CSS sieht das so aus:

```
@page {
  size: 338.67mm 190.5mm;
  margin: 0;
}

.folie {
  width: 338.67mm;
  height: 190.5mm;
  break-after: page;
  print-color-adjust: exact;
}
```

Eine Falle nebenbei: PowerPoint kennt zwei Einstellungen mit dem Verhältnis 16:9. „Bildschirmpräsentation (16:9)“ misst 10 mal 5,625 Zoll, „Breitbild“ misst 13,333 mal 7,5 Zoll. Gleiches Verhältnis, unterschiedliche Fläche. Microsoft empfiehlt Breitbild, weil mehr Platz für Inhalt bleibt.

Wenn es doch eine .pptx sein muss

Manchmal führt kein Weg daran vorbei, weil ein Kunde die Datei bearbeiten will. Dann gibt es einen dokumentierten Umweg über Marp, ein Open-Source-Werkzeug, das Markdown in HTML, PDF und PowerPoint umwandelt und dafür intern einen Chromium-Browser nutzt.

```
npx @marp-team/marp-cli@latest deck.md -o deck.pptx
```

Sei dir dabei über eine Einschränkung im Klaren, und die steht so in der offiziellen Dokumentation: Das erzeugte PPTX besteht normalerweise aus vorgerenderten Hintergrundbildern. Die Inhalte lassen sich in PowerPoint also nicht bearbeiten oder weiterverwenden. Es gibt zusätzlich eine Option für editierbare PPTX-Dateien, die ist aber ausdrücklich als experimentell markiert, braucht LibreOffice Impress als zusätzliche Installation, unterstützt keine Sprechernotizen, und das Marp-Team schreibt selbst, dass es davon abrät, wenn das Aussehen der Folien wichtig ist.

Meine ehrliche Empfehlung: Wenn wirklich jemand die Folien bearbeiten muss, bau sie von Anfang an in PowerPoint. Wenn nicht, bleib bei HTML und schick ein PDF.

Ein Wort zu Online-Convertern

Es gibt viele Seiten, die HTML in PDF umwandeln, kostenlos und in zehn Sekunden. Für einen Blogbeitrag ist das in Ordnung. Für ein Pitch-Deck mit Umsatzzahlen, für eine interne Strategie oder für Kundenmaterial lädst du damit vertrauliche Informationen auf einen fremden Server, über dessen Umgang mit deinen Daten du nichts weißt.

Der Druckdialog in deinem eigenen Browser kostet dich zwanzig Sekunden mehr und verlässt deinen Rechner nicht.

Kapitel 6: Die Fehler, die dich Zeit kosten

Du lässt bauen, bevor gedacht ist. Der teuerste Fehler von allen. Wenn du die Struktur überspringst, korrigierst du später am fertigen Design herum, statt vorher zwei Sätze umzustellen.

Du gibst Geschmack als Gefühl weiter. „Mach es hochwertiger" ist keine Anweisung, das ist ein Wunsch. „Headline 44 Punkt statt 36, Abstand nach unten verdoppeln, maximal drei Bullets pro Folie" ist eine Anweisung.

Du sammelst deine Korrekturen nicht ein. Wenn du beim fünften Deck dieselben drei Dinge korrigierst wie beim ersten, arbeitest du für die KI statt mit ihr. Zurückschreiben in die Guideline, immer.

Du denkst den Druck erst am Ende mit. Das Deck sieht im Browser perfekt aus, und im PDF ist plötzlich alles verrutscht. Druckregeln gehören ins Fundament, nicht in die Nachbesserung.

Du lässt zu viel Text auf eine Folie. Die KI wird versuchen, dir zu gefallen, und die Schrift verkleinern, bis alles passt. Dann steht 14-Punkt-Text auf einer Folie, die aus zehn Metern Entfernung gelesen werden soll. Gib in den Prompt hinein, dass sie bei Platzmangel meckern soll, statt zu schrumpfen.

Du prüfst die Zahlen nicht. KI erfindet Zahlen, wenn du sie nicht lieferst, und sie erfindet sie überzeugend. Jede Zahl auf einer Folie muss aus deiner Quelle stammen. Jede. Wenn dich jemand im Termin auf eine Zahl festnagelt, die die KI sich ausgedacht hat, ist der Rest der Präsentation egal.

Du baust jedes Mal ein neues Designsystem. Der Punkt des ganzen Vorgehens ist die Wiederverwendung. Eine Guideline, viele Decks. Wenn du für jedes Thema eine neue Referenz zerlegst, hast du den Aufwand ohne den Ertrag.

Du erwartest, dass es beim ersten Mal schneller geht. Tut es nicht. Der erste Durchlauf kostet dich einen halben Tag, weil du die Guideline aufbaust. Ab dem dritten Deck holst du das zurück. Wer das nicht weiß, hört nach dem ersten Versuch auf und erzählt danach, dass KI für Präsentationen nichts taugt.

Kapitel 7: Checkliste

Vor dem Start

- [ ] Designreferenz ausgewählt, die die gewünschte Wirkung trifft
- [ ] Guideline als Markdown-Datei abgelegt, mit CSS-Tokens am Ende
- [ ] Zielgruppe in ganzen Sätzen beschrieben, nicht als Schlagwort
- [ ] Ziel formuliert: Was soll nach dem Termin passieren
- [ ] Eigene Zahlen, Quellen und Texte gesammelt und bereitgelegt

Struktur

- [ ] Kernbotschaft steht in einem Satz
- [ ] Dramaturgie in 5 bis 7 Stationen
- [ ] Jede Folie hat eine benannte Funktion im roten Faden
- [ ] Folien ohne Funktion sind gestrichen
- [ ] Keine Folie hat mehr als eine Kernaussage
- [ ] Alle Zahlen haben eine Quelle, nichts steht als Platzhalter drin

Bau

- [ ] Die KI hat einen Vorab-Plan geliefert, bevor sie Code geschrieben hat
- [ ] Design-Tokens stehen als CSS-Variablen an einer Stelle
- [ ] Druckregeln stehen im Fundament, nicht nachträglich
- [ ] Eine Datei, alles inline, keine externen Abhängigkeiten
- [ ] Seitenumbruch nach jeder Folie gesetzt

Prüfung

- [ ] Kein Text läuft über den Folienrand
- [ ] Schriftgrößen sind aus zehn Metern Entfernung lesbar
- [ ] Abstände sind über alle Folien gleich
- [ ] Zahlen und Namen gegen die Originalquelle geprüft
- [ ] Umlaute und Sonderzeichen werden korrekt dargestellt
- [ ] Deck einmal komplett laut durchgesprochen, nicht nur angeschaut

Export

- [ ] PDF erzeugt, Querformat, Ränder auf Keine
- [ ] Hintergrundgrafiken sichtbar
- [ ] Jede Folie liegt auf genau einer Seite, keine leeren Seiten
- [ ] PDF auf einem zweiten Gerät geöffnet und geprüft
- [ ] Dateiname trägt Datum und Version

Danach

- [ ] Korrekturen als Regeln in die Guideline zurückgeschrieben
- [ ] Guideline versioniert abgelegt, damit der nächste Start schneller geht

Kapitel 8: Was das Ganze wirklich bringt

Ich schreibe hier bewusst keine Prozentzahl hin, wie viel Zeit du sparst. Solche Zahlen kursieren, aber ich habe keine belastbare Quelle dafür gefunden, und ich schreibe nichts hin, was ich nicht belegen kann.

Was ich dir sagen kann, ist die Struktur der Ersparnis, und die ist wichtiger als eine Prozentzahl.

Der Aufwand verschiebt sich. Nach vorne, in das Aufschreiben deines Geschmacks. Und nach hinten, in die Prüfung, ob das Ergebnis wirkt. Der Teil dazwischen, das Ausformulieren, das Layouten, das Ausrichten von 40 Textkästen, verschwindet weitgehend.

Genau dieser mittlere Teil ist der, den niemand von uns gerne macht und der trotzdem die meisten Stunden frisst. Der Teil, den du behältst, ist der, für den man dich eigentlich bezahlt: zu wissen, was gesagt werden muss, und zu erkennen, ob es angekommen ist.

Wenn du mit KI Folien bauen willst, hör auf, in PowerPoint zu denken. Die Sprache der Modelle ist Code, und dein Job ist es, ihnen beizubringen, was du für gut hältst.

Und dann hör bei den Folien nicht auf. Das Muster aus diesem E-Book, Geschmack aufschreiben, die Maschine im richtigen Format bauen lassen, Korrekturen in eine Guideline zurückschreiben, funktioniert genauso für ein Video, für eine Landingpage, für vertonte Texte, für eine ganze Kette von Aufgaben, die dir bisher Stunden weggenommen hat. Präsentationen sind die kleinste Übung davon. Sie sind nur die, an der man es am schnellsten lernt.

Quellen und weiterführende Links

Jede technische Aussage in diesem E-Book ist an der Primärquelle geprüft. Hier sind die Belege, damit du selbst nachlesen kannst.

Technische Belege

- Microsoft, Aufbau eines PresentationML-Dokuments (warum eine .pptx ein Paket aus XML-Teilen ist): https://learn.microsoft.com/en-us/office/open-xml/presentation/structure-of-a-presentationml-document
- Microsoft, Foliengrößen in PowerPoint (Breitbild 13,333 mal 7,5 Zoll): https://support.microsoft.com/en-us/powerpoint/change-the-size-of-your-powerpoint-slides
- MDN, CSS-Regel @page (Seitengröße, Ausrichtung, Ränder im Druck): https://developer.mozilla.org/en-US/docs/Web/CSS/@page
- MDN, print-color-adjust (warum Hintergründe beim Drucken verschwinden): https://developer.mozilla.org/en-US/docs/Web/CSS/print-color-adjust
- reveal.js, PDF-Export (Druckeinstellungen für Folien-Decks): https://revealjs.com/pdf-export/
- Marp, Markdown zu HTML, PDF und PowerPoint: https://marp.app/
- Marp CLI, Dokumentation zum PPTX-Export und seinen Grenzen: https://github.com/marp-team/marp-cli
- Claude-Dokumentation zu Skills (SKILL.md als wiederverwendbare Arbeitsanweisung): https://code.claude.com/docs/en/skills
- Claude-Support zu Artifacts (Vorschau von einseitigem HTML): https://support.claude.com/en/articles/9487310-what-are-artifacts-and-how-do-i-use-them

Stand der Prüfung aller Links und Angaben: Juli 2026. Preise und Funktionsumfänge von KI-Werkzeugen ändern sich schnell. Deshalb stehen in diesem E-Book bewusst keine Anbieterpreise. Prüf sie beim Anbieter selbst, bevor du eine Entscheidung triffst.

Über den Autor

Maik Schwede ist Unternehmer. Er hat ein Unternehmen aufgebaut, es verloren, und danach wieder eines aufgebaut. Bei swing2sleep hat er eine Produktkategorie mit entwickelt, die inzwischen mehrere hunderttausend Familien erreicht hat.

Heute arbeitet er als Sparringspartner für Unternehmer, die an einem Punkt stehen, an dem der nächste Schritt nicht offensichtlich ist. Kein Coach, kein Guru. Jemand, der die Lage von innen kennt und mitdenkt.

Sein eigener Arbeitsalltag läuft mit KI-Systemen, die er selbst aufgesetzt hat und die er per Sprachnachricht steuert. Damit entstehen Websites, Videos, vertonte Artikel und ganze Arbeitsabläufe, die ohne ihn weiterlaufen. Dieses E-Book zeigt einen kleinen Ausschnitt davon.

Wenn du an einem Thema festhängst und einen ehrlichen zweiten Kopf brauchst, findest du auf **maikschwede.de** die Möglichkeit, ein Erstgespräch zu buchen. Dreißig Minuten reichen meistens, um die Stelle zu finden.