PhantomWP Cheat Sheet

Dieses Cheat Sheet basiert auf den bereitgestellten Docs aus **Docs_small.pdf** und ist so aufgebaut, dass du **möglichst schnell produktiv** mit **PhantomWP** arbeiten kannst.

---

1. Was ist PhantomWP?

PhantomWP ist ein **headless Site Builder**, der deine **WordPress-Seite** in eine **schnelle statische Astro-Seite** verwandelt.

Du behältst:

- **WordPress** für Content Management
- die gewohnte Redaktion
- bestehende Inhalte

Du bekommst zusätzlich:

- **Astro-Frontend**
- **statische HTML-Ausgabe**
- **mehr Performance**
- **mehr Sicherheit**
- **weniger Wartung**
- **günstiges oder kostenloses Hosting**

Kurz gesagt

WordPress bleibt dein CMS.
PhantomWP baut daraus eine moderne statische Website.

---

2. Warum PhantomWP statt klassischem WordPress-Frontend?

Die Probleme von normalem WordPress

Laut Doku bringt klassisches WordPress oft diese Nachteile:

- **langsame Seitenladezeiten**
  - PHP-Verarbeitung
  - Datenbankqueries auf jedem Page Load
- **höhere Hosting-Kosten**
  - PHP-Hosting
  - Datenbank
  - oft zusätzlich Caching
- **laufende Wartung**
  - Core-Updates
  - Theme-Updates
  - Plugin-Updates
- **Sicherheitsrisiken**
  - Plugins
  - PHP-Angriffsfläche
  - Datenbank-Angriffsfläche

Was PhantomWP dagegen macht

PhantomWP generiert **statische HTML-Dateien** aus deinen WordPress-Inhalten.

Vorteile:

- **lädt sofort**

- **kein PHP zur Laufzeit**
- **keine DB zur Laufzeit**
- **inhärent sicherer**
- **minimaler Wartungsaufwand**
- **kostenlos hostbar** auf z. B.:
  - Vercel
  - Netlify
  - anderen Static Hosts

---

3. Für wen ist PhantomWP gedacht?

PhantomWP ist laut Doku ideal für:

- **WordPress-Site-Owner**, die bessere Performance wollen
- **Content Creators**, die weiter mit WordPress schreiben möchten
- **Agenturen**, die mehrere WordPress-Seiten betreuen
- **Entwickler**, die modernen Stack wollen ohne Migrationshölle
- **alle**, die WordPress-Sicherheitsupdates und langsame Ladezeiten satt haben

---

4. Das Grundprinzip / der Workflow

Der Kern-Workflow ist extrem einfach:

1. **Content in WordPress schreiben**
2. **Design und Frontend in PhantomWP/Astro bauen**
3. **statisch deployen**

Formel:

WordPress (Content) → PhantomWP (Build) → Astro Site (Deploy)

---

5. Was du in PhantomWP bekommst

5.1 Flexible Development Environments

Du kannst Projekte in drei Modi laufen lassen:

GitHub Codespaces

- Cloud-Entwicklungsumgebung
- im Browser nutzbar
- nichts lokal installieren

Docker Local Development

- lokal auf deinem Rechner
- ideal für Offline-Arbeit
- gleicher IDE-Flow wie in der Cloud

Fly.io Remote Containers

- eigener immer laufender Cloud-Container

Wichtig

Alle drei Modi teilen sich laut Doku:

- gleiche IDE
- gleicher AI Assistant
- gleiche WordPress-Integration
- gleiche Deployment-Flows

5.2 Web-Based IDE

Die IDE enthält:

- **Monaco Editor**
- **Syntax Highlighting**
- **IntelliSense**
- **Live Preview**
- **Visual Editor**
- **AI Assistant**
- **Git Integration**

5.3 WordPress Integration

Du kannst dein bestehendes WordPress nutzen zum:

- Inhalte durchsuchen
- Inhalte importieren
- WordPress als Headless CMS verwenden

5.4 Pre-Built Components

Component Library mit fertigen Sektionen, z. B.:

- Hero Sections
- Feature Grids
- Testimonials
- Pricing Tables
- Contact Forms

5.5 One-Click Deployment

Deployment auf **Vercel** mit:

- automatischen Builds
- globalem CDN
- HTTPS
- Preview Deployments pro Branch

---

6. Schnellstart in 5 Minuten

Voraussetzungen

Du brauchst nur:

- einen **GitHub-Account**
- eine **E-Mail-Adresse**
- etwa **5 Minuten**
- **keine lokale Installation**
- **keine CLI-Kenntnisse**

Schritt 1: Account erstellen

1. Zu **PhantomWP** gehen
2. **Get Started** klicken
3. Mit E-Mail registrieren
4. Mail bestätigen

Schritt 2: GitHub verbinden

1. Im Dashboard **Connect GitHub Account**
2. GitHub-Zugriff autorisieren
3. Zurück ins Dashboard

Wichtig

PhantomWP braucht GitHub-Zugriff, um:

- Repositories zu erstellen
- Codespaces zu verwalten

Schritt 3: Erstes Projekt erstellen

1. **Create New Project**
2. Repo-Name eingeben, z. B. `my-astro-site`
3. Public oder Private wählen
4. Laufzeit wählen:
   - GitHub Codespace
   - Docker
   - Fly.io
5. Starten:
   - **Create & Launch Codespace**
   - oder lokal / Fly.io

PhantomWP macht dann automatisch:

- neues GitHub-Repo
- Astro-Template einrichten
- Codespace starten
- IDE öffnen

Dauer: ca. **3–4 Minuten**

Schritt 4: IDE verstehen

Die wichtigsten Bereiche:

Bereich	Zweck
File Tree (links)	Dateien durchsuchen/verwalten
Code Editor (Mitte)	Code bearbeiten
Live Preview (rechts)	Änderungen sofort sehen
Header Bar (oben)	Tools und Einstellungen

Schritt 5: Erste Änderung

1. `src/pages/index.astro` öffnen
2. `<h1>` suchen
3. Text ändern
4. `Cmd/Ctrl + S`

Die Preview aktualisiert sich sofort.

Schritt 6: Komponente einfügen

1. **Sections-Icon** im Header klicken
2. Sektion auswählen
3. Vorschau ansehen
4. **Insert**

Schritt 7: Deployen

Änderungen committen

1. Git-Icon klicken
2. Änderungen prüfen
3. Commit-Message schreiben
4. **Commit & Push**

Vercel einrichten

1. Im Dashboard Projekt finden
2. **Setup Deployment**
3. Vercel-Token eingeben
4. Repo verbinden

Danach wird **jeder Push automatisch deployed**.

---

7. Das Wichtigste zuerst: dein praktischer Minimal-Workflow

Wenn du **sofort loslegen** willst, dann arbeite erstmal nur so:

Workflow A: Neue Marketing-Seite bauen

1. Projekt erstellen
2. `src/pages/index.astro` anpassen
3. Component Library öffnen
4. Hero + Features + CTA einfügen
5. Theme Studio öffnen
6. Farben + Fonts setzen
7. Git committen
8. Vercel deployen

Workflow B: WordPress-Content headless nutzen

1. WordPress verbinden
2. Content-Typen auswählen
3. **Generate Pages**
4. `src/lib/wordpress.ts` nutzen
5. Blog-Listing + Detailseiten bauen
6. Media runterladen
7. Deployen

---

8. IDE Cheat Sheet

8.1 Header Bar – was ist wo?

Die Toolbar ist laut Doku in Bereiche gegliedert:

Context & Status

- aktuelle Datei
- Verbindungsstatus
- Services

Layout Panels

- File Tree ein-/ausblenden
- Editor
- Preview
- Structure
- AI Chat

Design Tools

- Sections Library
- Icons
- Fonts
- Tokens / Theme
- WordPress
- SEO
- Menus
- Media

Git & Deploy

- File History
- Änderungen

- Commit
- Deploy

Navigation

- More
- Help
- Dashboard

8.2 File Tree – wichtigste Ordner

Ordner	Zweck
`src/pages/`	Seiten / Routen
`src/content/`	Content / Blog-Inhalte
`src/components/`	wiederverwendbare Komponenten
`src/layouts/`	Layouts
`public/`	statische Assets
`src/media/`	optimierte Bilder

8.3 Editor-Funktionen

Der Editor bietet:

- Syntax Highlighting
- IntelliSense
- Fehlerdiagnosen inline
- Multiple Tabs
- Code Folding
- Find/Replace
- Regex-Suche

8.4 Wichtige Shortcuts

Speichern / Suchen / Formatieren

Aktion	Mac	Windows
Save	`Cmd+S`	`Ctrl+S`
Find	`Cmd+F`	`Ctrl+F`
Replace	`Cmd+H`	`Ctrl+H`
Go to Line	`Cmd+G`	`Ctrl+G`
Toggle Comment	`Cmd+/`	`Ctrl+/`
Format Document	`Shift+Option+F`	`Shift+Alt+F`

Weitere wichtige

Aktion	Mac	Windows
Quick Open	`Cmd+P`	`Ctrl+P`
New File	`Cmd+N`	`Ctrl+N`
Save All	`Cmd+Option+S`	`Ctrl+Alt+S`
Close Tab	`Cmd+W`	`Ctrl+W`
Go to Symbol	`Cmd+Shift+O`	`Ctrl+Shift+O`

8.5 Preview

Die Preview:

- refresht automatisch beim Speichern
- erlaubt responsive Testing
- hat manuellen Refresh-Button
- zeigt die Seite wie für Besucher

---

9. Working with Files – Seiten, Komponenten, Blogposts, Layouts

---

9.1 Seiten erstellen

Wege

- `+ Page` Button
- Hover auf `src/pages` → `+`
- Rechtsklick → New Page

Templates

- Blank Page
- Contact Page
- About Page
- FAQ Page
- 404 Error Page

Formate

- `.astro` → volle Kontrolle
- `.mdx` → Markdown + Komponenten

Layouts

- `BaseLayout`
- `PageLayout`
- `ContentLayout`

9.2 URL-Mapping verstehen

Dateipfad	URL
`src/pages/index.astro`	`/`
`src/pages/about.astro`	`/about`
`src/pages/services/design.astro`	`/services/design`
`src/pages/blog/my-post.mdx`	`/blog/my-post`

9.3 Komponenten erstellen

Ort: `src/components/`

Wege

- `+ Component`
- Hover auf `src/components`
- Rechtsklick → New Component

Templates

- Simple Component
- Component with Props
- Card Component
- Button Component
- Hero Section

Props-Beispiel

```
---
interface Props {
  title: string;
  description?: string;
}
```

```
const { title, description } = Astro.props;
---
<div class="card">
  <h3>{title}</h3>
  {description && <p>{description}</p>}
</div>
```

Verwendung

```
---
import Card from '../components/Card.astro';
---
<Card title="My Card" description="Card content here" />
```

9.4 Komponenten extrahieren

Im HTML Tree View:

1. Element auswählen
2. Rechtsklick
3. **Extract Component**
4. Namen eingeben

PhantomWP macht automatisch:

- neuen File in `src/components/`
- Originalcode ersetzen durch `<ComponentName />`
- Import hinzufügen

9.5 Blogposts erstellen

Ort: `src/pages/blog/`

Format: **MDX**

Frontmatter-Beispiel

```
---
title: "My Blog Post"
description: "A short description"
pubDate: 2024-01-15
author: "Your Name"
tags: ["tutorial", "astro"]
---
# Introduction
Write your content in Markdown...
```

Typische Frontmatter-Felder

Feld	Zweck
`title`	Titel
`description`	SEO-Text
`pubDate`	Veröffentlichungsdatum
`author`	Autor
`tags`	Tags
`image`	Featured Image
`draft`	`true`, um Post zu verstecken

9.6 Layouts erstellen

Ort: `src/layouts`

Beispiel

```
---
interface Props {
  title: string;
  description?: string;
```

```
}
const { title, description } = Astro.props;
---
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width" />
  <title>{title}</title>
  <meta name="description" content={description} />
</head>
<body>
  <header></header>
  <slot />
  <footer></footer>
</body>
</html>
```

Nutzung

```
---
import BaseLayout from '../layouts/BaseLayout.astro';
---
<BaseLayout title="My Page">
  <main>
    <!-- content -->
  </main>
</BaseLayout>
```

9.7 Dateien verwalten

Umbenennen

- Rechtsklick → Rename
- oder Pencil-Icon

Doppeln

- Rechtsklick → Duplicate

Verschieben

- Drag & Drop

Löschen

- Rechtsklick → Delete

Ordner anlegen

- Rechtsklick auf Parent → New Folder

Suchen

- `/` drücken oder Suchsymbol

---

10. Visual Editor / HTML Tree View

Der Visual Editor ist extrem nützlich, wenn du nicht alles direkt im Code suchen willst.

10.1 Öffnen

- Rechts im Panel den **Tree**-Tab öffnen

10.2 Was du dort siehst

Die Baumansicht zeigt:

- HTML-Tags
- Astro-Komponenten
- Loops
- Conditionals
- Klassen
- IDs
- Textinhalt
- Section Names via HTML-Kommentare

10.3 Drei-Wege-Sync

Wenn du ein Element im Tree auswählst:

1. Element wird in der Preview hervorgehoben
2. Editor springt zur Code-Stelle
3. Properties Panel zeigt bearbeitbare Eigenschaften

Das ist super zum Debuggen.

10.4 Eigenschaften bearbeiten

Classes

- Klassen bearbeiten
- Tailwind-Autocomplete
- Klasse temporär deaktivieren
- Klasse kopieren
- Klasse löschen

ID

- direkt ändern

Component Props

- definierte Props sehen
- Werte anpassen
- Booleans toggeln
- neue Props hinzufügen

10.5 Drag & Drop

Du kannst HTML-Elemente und Komponenten umsortieren.

Nicht verschiebbar:

- Loop-Elemente
- Conditional-Elemente

Beim Drop siehst du Indikatoren:

- blaue Linie oben → davor
- blaue Linie unten → danach
- blauer Hintergrund → als Kind

10.6 Context Menu im Tree

Mögliche Aktionen:

- Insert Before
- Insert After
- Insert Child
- Add/Edit Section Name
- Extract Component
- Rename Component
- Delete Element

10.7 Beste Anwendung

Nutze den Tree besonders für:

- Styling-Fehler finden
- DOM-Struktur verstehen
- komplexe Seiten durchblicken
- richtige Stelle im Code finden
- Tailwind-Klassen testen

---

11. Menu Builder

Mit dem Menu Builder erzeugst du **wiederverwendbare Astro-Menü-Komponenten**.

11.1 Öffnen

- **Menu-Icon** in der Toolbar

11.2 Layout des Builders

Bereich	Zweck
links	Menüliste + WordPress-Import
Mitte	Quellen für Menüpunkte
rechts	Menüstruktur + Einstellungen

11.3 Neues Menü erstellen

1. **New Menu**
2. Namen eingeben, z. B. `Main`

Dann entsteht z. B.:

- Komponente: `<MainMenu />`
- Datei: `src/components/menus/MainMenu.astro`

11.4 Zwei Betriebsmodi

Auto-sync Mode

Empfohlen.

- entdeckt automatisch Seiten aus `src/pages/`
- aktualisiert sich bei neuen/gelöschten Seiten
- Seiten können ausgeschlossen werden
- Reihenfolge per Drag & Drop

Manual Mode

- du fügst alle Items manuell hinzu
- volle Kontrolle
- nichts aktualisiert sich automatisch

11.5 Menüpunkte hinzufügen

Aus lokalen Seiten

- einzelne Seiten
- oder **Add All Pages**

Aus WordPress

- wenn verbunden
- published WP-Seiten werden gezeigt

Custom Links

- Label + URL
- für externe Links oder Spezialseiten

11.6 Dropdowns bauen

Im Manual Mode:

1. Parent-Item anlegen
2. Child-Items darunter
3. mit **Indent** einrücken

11.7 Menüstile

- Horizontal
- Vertical
- Dropdown

Optional:

- Hamburger-Menü aktivieren

11.8 WordPress-Menüs importieren

Wenn WordPress verbunden ist und dort Menüs definiert sind:

1. unten links unter „Import from WordPress“
2. Menü anklicken
3. Struktur wird in Astro-Menü kopiert

11.9 Speichern und nutzen

Beim Speichern passiert:

- Astro-Komponente wird erzeugt/aktualisiert
- Konfiguration wird in `src/config/menus.json` gespeichert

Nutzung im Layout

```
---
import MainMenu from '../components/menus/MainMenu.astro';
---
<header>
  <MainMenu />
</header>
```

---

12. Component Library

Die Component Library ist dein Beschleuniger für schnelles Bauen.

12.1 Öffnen

- **Sections-Icon** in der Toolbar

12.2 Tabs

Tab	Zweck
Sections	einzelne Sektionen nach Kategorie
Quick Start	fertige Bundles für Seitentypen

12.3 Kategorien

- Hero
- Features
- Testimonials
- Pricing
- CTA
- Contact
- Navigation
- Footer
- Content
- Gallery
- Stats
- Team
- FAQ
- Logos

12.4 Vorschau

Du kannst Sektionen ansehen mit:

- Live Preview
- Theme Colors
- Code View
- Copy Code

Verfügbare Theme-Farben:

- Indigo
- Blue
- Emerald
- Rose
- Amber
- Violet

12.5 Einfügen

Einzelne Sektion

1. auswählen

2. previewen
3. **Insert Component**

Mehrere Sektionen

1. Checkboxen aktivieren
2. Page View anschauen
3. **Insert All**

Einfüge-Reihenfolge = Auswahl-Reihenfolge.

12.6 Quick Start Templates

Vorgefertigte Seiten-Bundles, z. B.:

- Landing Page
- SaaS Landing
- Portfolio
- About Page
- Pricing Page
- Freelancer
- Product Landing
- Agency Website

Perfekt, wenn du in Minuten eine komplette Startseite brauchst.

12.7 Nach dem Einfügen anpassen

Typische Anpassungen:

- Texte ändern
- Bilder austauschen
- Farben anpassen
- Karten/Testimonials duplizieren oder entfernen

12.8 AI dafür nutzen

Beispielprompts:

- „Change the background to a gradient“
- „Add a fourth pricing tier“
- „Make this section full-width“

---

13. Media Manager

Bilder werden in PhantomWP sehr komfortabel verwaltet.

13.1 Zwei Wege

Media Library

Visuelle Galerie

File Tree

Dateibasierte Verwaltung

13.2 Optimierte und statische Medien

Pfad	Zweck
`src/media/`	optimierte Bilder via Astro
`public/`	statische Dateien ohne Optimierung

`src/media/` – empfohlen für Bilder

Vorteile:

- Formatkonvertierung
- responsive Größen
- lazy loading
- blur placeholders

`public/`

für z. B.:

- favicon
- robots.txt
- Open Graph Defaults
- Dateien, die exakt so bleiben müssen

13.3 Unterstützte Formate

- JPG
- JPEG
- PNG
- GIF
- WebP
- SVG
- AVIF

13.4 Bilder einfügen

Hover auf Bild → Aktionen:

- Insert at Cursor
- Copy Import
- Copy Full Code
- Delete

Eingefügter Code

```
---
import { Image } from 'astro:assets';
import heroImage from '../media/hero.jpg';
---
<Image src={heroImage} alt="hero" />
```

13.5 Drag & Drop in den Editor

Du kannst Bilder aus `src/media/` direkt in den Code ziehen.
PhantomWP fügt dann Import + `Image`-Komponente automatisch ein.

13.6 Best Practices für Medien

Dateinamen

Gut:

- `team-photo-2024.jpg`
- `product-dashboard.png`

Schlecht:

- `IMG_2847.jpg`
- `Screenshot 2024-01-15.png`

Ordnerstruktur

z. B.

- `src/media/blog/`
- `src/media/products/`
- `src/media/team/`
- `src/media/icons/`

Bildgrößen

- Hero: max. 1920px
- Content: 800–1200px
- Thumbnails: ca. 400px
- Icons: idealerweise SVG

Alt-Texte

- beschreibend
- kurz
- nicht mit „Bild von …“ anfangen

---

14. Font Manager

Mit dem Font Manager installierst du Schriften aus **Fontsource**.

14.1 Öffnen

- **Type-Icon (T)**

14.2 Kategorien

- All Fonts
- Sans Serif
- Serif
- Monospace
- Display
- Handwriting

14.3 Features

- Suche
- Preview
- variable Fonts filtern
- Gewichte auswählen
- Fontsource-Link
- direkte Installation ins Projekt

14.4 Gewichte

Standardmäßig ausgewählt:

- `400`
- `700`

14.5 Installation

Add to Project installiert:

- `@fontsource/{font-id}`

Wichtig:

- Dev Server startet dabei neu
- Preview kann kurz leer sein

14.6 Aktivierung

Nach Installation musst du die Schrift in **Theme Studio** auswählen:

1. Theme Studio öffnen
2. Typography-Bereich
3. Font auswählen
4. **Apply Theme**

PhantomWP macht dann automatisch:

- `theme.css` updaten
- Font-Import in `BaseLayout.astro` einfügen

14.7 Direkte Nutzung

```
<h1 class="font-heading text-4xl">Welcome</h1>
<p class="font-sans">Body text uses the sans font.</p>
```

14.8 Best Practices

- max. 1–2 Fontfamilien
- nur benötigte Weights installieren
- möglichst variable Fonts nutzen

---

15. Theme Studio

Hier definierst du das visuelle System deiner Seite.

15.1 Öffnen

- **Palette-Icon**

15.2 Zweck

Theme Studio erstellt ein Tailwind-v4-`theme.css` mit Tokens für:

- Farben

- Typografie
- Spacing
- Border Radius
- Shadows

15.3 Aufbau

Panel	Inhalt
links	Editor
rechts	Live Preview

15.4 AI Theme Generation

Du kannst ein Theme mit Prompt erzeugen, z. B.:

- „Modern SaaS theme with teal and coral accents“

AI erzeugt dann:

- Farbpalette
- Light/Dark
- Font-Pairing

Wenn kein API-Key da ist, gibt es einen Fallback.

15.5 Presets

Es gibt 18 Preset-Themes, darunter:

Dark

- PhantomWP Default
- Midnight
- Dracula

- Nord
- Monokai
- Obsidian

Light

- Ocean Blue
- Forest
- Sunset
- Purple Haze
- Rose Gold
- Coral Reef
- Mint Fresh
- Lavender
- Amber Glow
- Steel
- Cherry
- Electric

15.6 Farbgruppen

Token-Gruppen:

- Primary
- Secondary
- Accent
- Surface
- Content
- Outline
- Status

Jede Farbe hat zugehörige Tailwind-Utilities wie:

- `bg-primary`
- `text-primary`

15.7 Light / Dark Mode

Jede Mode hat eigenes Token-Set.

Dark-Mode-Strategien

- **None** → nur Light
- **Class-based** → `.dark`
- **Media query** → `prefers-color-scheme`

15.8 Typography

Verfügbare Font-Token:

- `--font-sans`
- `--font-heading`
- `--font-mono`

Installierte Fonts erscheinen oben in den Dropdowns.

Standardfonts u. a.:

- Inter
- Space Grotesk
- Plus Jakarta Sans
- DM Sans
- Outfit
- Sora
- Manrope
- JetBrains Mono
- Fira Code

15.9 Spacing & Radius

Werden hauptsächlich als Referenz angezeigt.
Du nutzt im Code normale Tailwind-Klassen wie:

- `p-4`
- `gap-6`
- `rounded-lg`

15.10 Gespeicherte Themes / Import / Export

Du kannst:

- Theme lokal im Browser speichern
- Farben als JSON exportieren
- JSON oder CSS importieren

Wichtig:

- gespeicherte Themes liegen **im Browser Local Storage**
- nicht im Codespace
- nicht geräteübergreifend

15.11 Apply Theme

Beim Klick auf **Apply Theme** passiert:

1. `src/styles/theme.css` wird geschrieben
2. Font-Imports in `BaseLayout.astro` werden aktualisiert

15.12 Beispiel für Token-Nutzung

```
<section class="bg-surface p-8">
  <h2 class="font-heading text-3xl text-content">Welcome</h2>
  <p class="font-sans text-content-light">Body text with lighter color.</p>
  <button class="bg-primary text-white rounded-lg px-4 py-2 cursor-pointer hover:bg-primary-
dark">
    Get Started
  </button>
</section>
```

---

16. Icon Manager

Für Lucide-Icons direkt in Astro-Komponenten.

16.1 Öffnen

- **Shapes-Icon**

16.2 Vorteile

- 280+ Icons
- konsistenter Stil
- SVG-basiert
- tree-shakeable
- `@lucide/astro` ist bereits integriert

16.3 Kategorien

z. B.:

- Arrows & Navigation
- UI & Interface
- Actions
- Communication
- Media & Files
- Social
- Users & People
- Commerce
- Status & Feedback
- Security
- Time & Calendar
- Location & Maps
- Tech & Development
- Weather

- Objects
- Charts & Documents
- Text & Formatting
- Layout
- Shapes

16.4 Größen

Größe	Klassen	Pixel
Small	`w-4 h-4`	16x16
Medium	`w-6 h-6`	24x24
Large	`w-8 h-8`	32x32

16.5 Farben

Optionen z. B.:

- `currentColor`
- `text-primary`
- `text-white`
- `text-gray-400`
- `text-blue-500`
- `text-green-500`
- `text-red-500`
- `text-amber-500`
- `text-violet-500`

16.6 Import + Verwendung

Import

```
import ArrowRight from '@lucide/astro/icons/arrow-right';
```

Verwendung

```
<ArrowRight class="w-6 h-6" />
```

16.7 Beispiele

Basic

```
---
import Mail from '@lucide/astro/icons/mail';
---
<Mail class="w-5 h-5" />
```

Mit Farbe

```
<Check class="w-4 h-4 text-green-500" />
```

Icon Button

```
<button class="p-2 hover:bg-gray-100 rounded-lg cursor-pointer">
  <Settings class="w-5 h-5 text-gray-600" />
</button>
```

16.8 Accessibility

Bei Icon-only Buttons:

```
<button aria-label="Close menu" class="cursor-pointer">
  <X class="w-5 h-5" />
</button>
```

---

17. AI Assistant

Eines der stärksten Features in PhantomWP.

17.1 Zwei Modi

AI Chat Panel

für Gespräche, große Aufgaben, Projektänderungen

Inline AI

für schnelle lokale Code-Edits direkt im Editor

17.2 Modelle

Laut Doku u. a.:

- Claude Sonnet
- Claude Opus
- Claude Haiku
- GPT-5.2 Pro
- GPT-5.2
- Gemini 2.5 Pro
- Gemini 2.5 Flash
- Gemini 3 Pro
- Gemini 3 Flash

Wichtig

Du musst eigene API-Keys hinterlegen für:

- Anthropic

- OpenAI
- Google

17.3 Extended Thinking

Für komplexe Aufgaben: Brain-Icon aktivieren.

17.4 Was die AI tun kann

Tools:

- Create File
- Modify File
- Read File
- List Files
- Read Lines
- Search in File
- Get File Outline
- Restart Services
- Install Package
- Navigate Preview
- Get Documentation

Approval

Diese brauchen Freigabe:

- Create File
- Modify File
- Install Package

Optional:

- **Auto-approve**

17.5 Gute Beispielprompts

- „Create a testimonial card component with image and quote“
- „Add a navigation menu to the header“
- „Fix the mobile layout on this page“
- „Explain how this WordPress integration works“
- „Install the Astro sitemap integration“

17.6 Eigene AI-Instruktionen

Datei:
`docs/ai-instructions.md`

Beispiel:

```
# AI Instructions
## Preferences
- Use TypeScript for all new files
- Follow the existing component patterns
- Use Tailwind CSS for styling
- Keep components small and focused

## Project Context
- This is a marketing site for a SaaS product
- The main colors are blue-600 and gray-900
- We use the Inter font family
```

17.7 Inline AI

AI Modify

Ausgewählten Code ändern

Shortcut:

- `Cmd/Ctrl + Shift + M`

AI Generate

Neuen Code am Cursor erzeugen

Shortcut:

- `Cmd/Ctrl + Shift + G`

Submit

- `Cmd/Ctrl + Enter`

Close

- `Escape`

17.8 Tipps für bessere Resultate

Sei präzise

Schlecht:

- „Make a component“

Besser:

- „Create a Card component with an image at the top, a title, description, and a Learn More link. Use Tailwind CSS with rounded corners and a shadow.“

Kontext geben

Schlecht:

- „Fix this“

Besser:

- „The navigation menu doesn't show on mobile. The hamburger button appears but clicking it does nothing. How do I make it toggle?“

Fehlertext mitgeben

z. B. vollständige Fehlermeldung kopieren

---

18. Claude Code CLI

Wenn du lieber im Terminal mit Claude arbeitest.

18.1 Vorteile

- terminal-native
- voller Projektkontext
- agentisches Coding
- im Codespace nutzbar
- mit Claude-Abo inklusive

18.2 Terminal öffnen

- Terminal-Icon oben

18.3 Installation

Wenn nicht installiert:

- **Install Claude** im Terminal-Header

18.4 Starten

```
claude
```

18.5 Typische Nutzungen

Code verstehen

- „What files make up the homepage?“
- „Explain how the navigation component works“

Änderungen

- „Add a dark mode toggle to the header“
- „Fix the mobile layout on the footer“

Commands

- „Install the Astro sitemap integration“
- „Update all npm packages“

18.6 Wichtige Commands

Command	Zweck
`/help`	Hilfe
`/clear`	Verlauf leeren
`/compact`	Gespräch komprimieren
`/cost`	Token-Kosten
`/plugin`	Plugins verwalten
`/quit`	beenden

18.7 Ohne Permission Prompts

```
claude --dangerously-skip-permissions
```

Nur nutzen, wenn du wirklich bewusst volle Freigabe geben willst.

---

19. WordPress verbinden

Einer der wichtigsten Bereiche.

19.1 Öffnen

- WordPress-Icon in der IDE

19.2 Verbindung herstellen

1. WordPress-URL eingeben, z. B. `https://yoursite.com`
2. **Fetch**
3. PhantomWP erkennt `/wp-json` automatisch

19.3 Was danach sichtbar wird

- Site Name
- Description
- Link zur WP-Seite
- verfügbare Content-Typen

19.4 Unterstützte Content-Typen

- Posts
- Pages
- Media
- Categories
- Tags
- Authors
- Custom Post Types

19.5 Search Index

Optional:

- FuseJS-Suchindex für Client-Side Search generieren

19.6 Media Download

Du kannst WordPress-Medien herunterladen nach:

- `src/media/cms/`

Vorteile:

- schnellere Auslieferung
- Astro-Optimierung
- keine Abhängigkeit von Online-WP

19.7 Generate Pages

Beim Klick auf **Generate Pages** entstehen u. a.:

Typ	Generierte Dateien
Posts	`src/pages/blog/[slug].astro`, `src/pages/blog/index.astro`

Typ	Generierte Dateien
Pages	`src/pages/[slug].astro`
Categories	`src/pages/category/[slug].astro`, `src/pages/category/index.astro`
Tags	`src/pages/tag/[slug].astro`, `src/pages/tag/index.astro`
Authors	`src/pages/author/[slug].astro`, `src/pages/author/index.astro`
CPTs	`src/pages/{type}/[slug].astro`, `src/pages/{type}/index.astro`

Zusätzlich:

- `src/lib/wordpress.ts`
- `src/components/FeaturedImage.astro`
- `astro.config.mjs` mit Bilddomains

---

20. WordPress Data Browser

Sehr hilfreich zum schnellen Arbeiten mit WP-Daten.

20.1 Öffnen

- WordPress-Icon → **Browse Content**

20.2 Tabs

- Posts
- Pages
- Media
- Categories
- Tags
- Users
- CPTs
- Code Examples

20.3 Was du tun kannst

- Inhalte durchsuchen
- Einträge aufklappen
- Felder ansehen
- Astro-Snippets kopieren
- komplette Beispieltemplates kopieren

20.4 Typische Snippets

Feld	Snippet
ID	`{post.id}`
Title	`{post.title.rendered}`
Slug	`{post.slug}`
Content	`<div set:html={post.content.rendered} />`
Excerpt	`{post.excerpt.rendered}`
Date	`{new Date(post.date).toLocaleDateString()}`
Link	`{post.link}`
Featured Image	`{getFeaturedImageUrl(post)}`
Author	`{getAuthor(post)?.name}`

20.5 Code Examples

Es gibt komplette Beispiele für:

- Blog Post List
- Single Post Page
- Navigation Menu
- Category Archive
- Sidebar Widget
- Author Page
- Search Results

- Static Pages

---

21. WordPress Content in Astro fetchen

21.1 Grundidee

WordPress bleibt das CMS.
Astro zieht Inhalte via REST API.

21.2 Der Client

PhantomWP generiert:
`src/lib/wordpress.ts`

Typische Funktionen:

```
const posts = await getPosts();
const post = await getPost('my-post-slug');
const pages = await getPages();
const categories = await getCategories();
```

21.3 Blog-Index-Beispiel

```
---
import { getPosts } from '../lib/wordpress';
import Layout from '../layouts/Layout.astro';
const posts = await getPosts({ perPage: 10 });
---
```

```
<Layout title="Blog">
  <h1>Latest Posts</h1>
  <ul>
    {posts.map(post => (
      <li>
        <a href={`/blog/${post.slug}`}>
          {post.title.rendered}
        </a>
      </li>
    ))}
  </ul>
</Layout>
```

21.4 Single-Post-Beispiel

```
---
import { getPost, getAllPosts } from '../lib/wordpress';
import Layout from '../layouts/Layout.astro';

export async function getStaticPaths() {
  const posts = await getAllPosts();
  return posts.map(post => ({
    params: { slug: post.slug },
  }));
}

const { slug } = Astro.params;
const post = await getPost(slug);
---
<Layout title={post.title.rendered}>
  <article>
    <h1 set:html={post.title.rendered} />
    <div set:html={post.content.rendered} />
  </article>
</Layout>
```

21.5 REST API liefert u. a.

Posts

- Titel
- Content
- Excerpt
- Datum
- Autor
- Featured Image
- Categories
- Tags
- Custom Fields
- SEO-Daten

Pages

- Titel
- Content
- Template
- Parent
- Menu Order
- Custom Fields

Media

- Bildgrößen
- Alt-Texte
- Captions
- URLs

Taxonomies

- Categories
- Tags
- Custom Taxonomies

Custom Post Types

wenn `show_in_rest: true`

21.6 Featured Images

Mit `_embed` verfügbar.

Beispiel:

```
const featuredImage = post._embedded?.['wp:featuredmedia']?.[0];
```

Helper:

```
import { getFeaturedImageUrl } from '../lib/wordpress';
const imageUrl = getFeaturedImageUrl(post, 'large');
```

21.7 SEO-Daten

Wenn WordPress Yoast oder Rank Math nutzt:

Yoast

```
const seoData = post.yoast_head_json;
```

Rank Math

```
const title = post.rank_math_title;
const description = post.rank_math_description;
```

21.8 Build-Time vs Runtime

Static Generation

empfohlen

```
---
const posts = await getPosts();
---
```

SSR

für sehr häufig aktualisierte Inhalte

```
---
// astro.config.mjs: output: 'server'
const posts = await getPosts();
---
```

21.9 Performance-Tipps

1. `_fields` nutzen
2. große Collections paginieren
3. Responses cachen
4. `_embed` nutzen

Beispiel:

```
const posts = await getPosts({
  perPage: 10,
  fields: ['id', 'slug', 'title', 'excerpt', 'date'],
  embed: true,
});
```

---

22. Bild-Optimierung für WordPress-Medien

PhantomWP hat einen **zweistufigen Bild-Workflow**.

22.1 Stage 1: IDE Download

Wenn du in den WP-Settings auf **Download Media** klickst:

1. Bilder werden aus WordPress geladen
2. Originale landen in `src/media/cms/`
3. WebP-Versionen landen in `public/media/cms/`
4. URL-Mapping wird aktualisiert

Das gibt dir sofort lauffähige Bilder in der Dev Preview.

22.2 Stage 2: Build-Time Sync

Vor jedem Production-Build läuft `sync-media`.

Er macht:

1. Posts und Pages fetchen
2. alle genutzten Bilder finden
3. neue/geänderte Bilder laden
4. responsive WebP-Varianten erzeugen
5. Blur Placeholders erzeugen
6. Mappings updaten

22.3 Responsive Größen

Typische Varianten:

- 320w
- 640w
- 960w
- 1200w
- Full size

22.4 Blur Placeholders

Für Featured Images werden kleine, unscharfe Vorschaubilder erzeugt.

Vorteil

- schöneres Lazy Loading
- schneller wahrgenommener Seitenaufbau

Nicht für

- Priority Images (`priority={true}`)

22.5 Wichtige Dateien

- `src/lib/media-map.json`
- `src/lib/responsive-map.json`
- `src/lib/image-placeholders.json`
- `public/media/cms/...`

22.6 Manuell ausführen

```
node scripts/sync-media.mjs
```

---

23. CDN Mode

Wenn WordPress bereits Bilder über CDN ausliefert.

Verwenden wenn:

- deine WP-Seite schon CDN-URLs nutzt
- du Bilder **nicht lokal** ins Repo ziehen willst
- schnellere Builds / kleineres Repo wichtiger sind

Local Mode behalten wenn:

- du Bilder committen willst
- du die lokale Image-Mapping-Strategie willst
- kein CDN vorhanden ist

---

24. PhantomWP Connect Plugin

Sehr wichtig, wenn du **mehr als read-only** willst.

24.1 Wann brauchst du das Plugin?

Nicht zwingend, wenn du nur lesen willst.

Ja, wenn du willst:

- Schreibzugriffe auf WordPress
- agentische AI-Scaffolds

- JWT Visitor Auth
- WooCommerce Customer Flows
- signierte Zwei-Wege-Kommunikation

24.2 Was das Plugin bringt

- Self-Pairing
- signierte Kommunikation
- JWT-Authentifizierung
- AI-Write-Access
- verschlüsselte Speicherung
- WooCommerce-Flows

24.3 Installation

Empfohlen: aus der PhantomWP IDE

1. WordPress-Icon
2. Site-URL eingeben
3. **Install PhantomWP Connect plugin**
4. ZIP herunterladen
5. in WP hochladen
6. aktivieren
7. einmal WP-Admin besuchen

Beim ersten Admin-Aufruf paart sich das Plugin automatisch.

24.4 Woran erkennst du, dass es klappt?

In WordPress Admin

- Status: **Paired**
- Project ID
- Last check-in timestamp

In PhantomWP IDE

- Access Level: **Full Access**
- nicht nur **Read-only**

24.5 Was Full Access freischaltet

AI kann WordPress direkt ändern

Beispiele:

- CPTs anlegen
- Taxonomien anlegen
- Field Groups anlegen
- Draft-Posts erzeugen
- Seiten in WP anlegen

Visitor JWT Authentication

für Frontend-Routen wie:

- `/account/login`
- `/account/register`
- `/account/forgot-password`
- `/account/reset-password`

Signed Writes

Jede schreibende Anfrage ist signiert.

24.6 Wichtige Endpunkte

Unter:
`/wp-json/phantomwp/v1/`

Auth

- `POST /auth/token`
- `POST /auth/validate`
- `GET /auth/me`
- `POST /auth/forgot-password`
- `POST /auth/reset-password`

Scaffolding

- `GET /scaffold/capabilities`
- `POST /scaffold/post-type`
- `POST /scaffold/taxonomy`
- `POST /scaffold/field-group`
- `GET /scaffold/managed`

Pairing

- `POST /pair/bootstrap`
- `POST /pair/disconnect`

---

25. WooCommerce Setup

Die WooCommerce-Integration ist laut Doku **alpha**, aber funktional.

25.1 Voraussetzungen

- PhantomWP Connect Plugin installiert und gepairt
- WooCommerce aktiv auf derselben WP-Seite

25.2 Verbindung testen

Token-Endpunkt:

```
curl -X POST https://yoursite.com/wp-json/phantomwp/v1/auth/token \
  -H "Content-Type: application/json" \
  -d '{"username": "your-wp-username", "password": "your-wp-password"}'
```

Erwartet wird ein JSON mit:

- `success`
- `token`
- `user_email`
- Rollen

25.3 In PhantomWP verbinden

Beim Erstellen:

- **WooCommerce Store** als Template wählen

Dann:

- Cart-Icon in der IDE
- WooCommerce Settings öffnen
- Felder konfigurieren:
  - WooCommerce URL
  - Access Secret
  - Payment Keys optional

Danach

- **Test Connection**
- **Save Configuration**

25.4 Produkte und Bilder synchronisieren

Sync Products & Images macht:

1. Produktdaten downloaden
2. Produktbilder lokal speichern
3. Media Map erzeugen

Dateien:

- `src/data/products.json`
- `src/media/products/`

25.5 Payments

Stripe

- Secret Key
- Publishable Key
- Webhook Secret

PayPal

- Client ID
- Client Secret
- Sandbox oder Live

Ohne Stripe/PayPal testen

In WooCommerce:

- **Direct bank transfer (BACS)** aktivieren

Dann kannst du den kompletten Checkout testen, ohne externe Zahlungsanbieter.

25.6 Relevante Env-Variablen

Automatisch in `.env`:

- `WP_API_URL`
- `WC_API_URL`
- `WP_ACCESS_SECRET`
- `JWT_SECRET`
- `PUBLIC_SITE_URL`
- ggf. Stripe/PayPal-Keys

---

26. WooCommerce Store Features

Generierte Seiten

- `/`
- `/shop`
- `/shop/product/[slug]`
- `/shop/category/[slug]`
- `/cart`
- `/checkout`
- `/order-complete`
- `/search`
- `/login`
- `/register`
- `/forgot-password`
- `/reset-password`
- `/account`
- `/account/orders`
- `/account/addresses`
- `/account/settings`

Architektur

Produktseiten

- Store API Daten bei Dev/Build-Zeit

Cart

- client-side mit `nanostores`
- `localStorage`

Checkout / Orders / Auth

- serverseitige API-Routes
- Verbindung via PhantomWP Connect

Wichtige Daten-Dateien

- `src/data/products.json`
- `src/data/categories.json`
- `src/media/products/`
- `src/lib/product-media-map.json`

Wichtige Helper

Produktdaten

- `getLocalProducts`
- `getLocalProduct`
- `getLocalCategories`
- `getLocalVisibleCategories`
- `getLocalProductsByCategory`

- `getLocalFeaturedProducts`
- `getLocalRelatedProducts`

WooCommerce Utilities

- `formatPrice`
- `isInStock`
- `getDiscountPercentage`
- `stripHtml`

Cart

- `$cart`
- `$cartCount`
- `$cartTotal`
- `addToCart`
- `removeFromCart`
- `updateQuantity`
- `clearCart`

Auth

- `$isLoggedIn`
- `$user`
- `$token`
- `login`
- `register`
- `logout`
- `getToken`
- `initAuth`
- `requestPasswordReset`

---

27. Docker Local Development

Wenn du lokal statt in Codespaces arbeiten willst.

27.1 Vorteile

- keine Codespace-Kosten
- offline nutzbar, sobald Projekt lokal ist
- gleiche IDE
- Dateien optional direkt lokal im Dateisystem
- gut für VS Code / Cursor / Claude Code

27.2 Architektur

Im Browser läuft weiter `phantomwp.com`,
aber lokal laufen im Container:

- Astro Dev Server
- WebSocket Server

27.3 Voraussetzungen

- Docker Desktop
- PhantomWP Account
- moderner Browser

27.4 Erster Start

Im Dashboard:

- **Run Locally** / **Start Locally**

Dann bekommst du einen `docker run` Befehl.

Beispiel:

```
docker run --name phantomwp-my-site \
  -p 14321:4321 \
  -p 14322:8080 \
  -v phantomwp-my-site:/app \
  -e PHANTOMWP_TOKEN=<your-token> \
  ghcr.io/phantomwp/local-dev:<release-tag>
```

27.5 Was der Container dann macht

- Image ziehen
- Projekt holen
- Git-Remote konfigurieren
- Abhängigkeiten installieren
- Astro + WebSocket starten

27.6 Danach

Im Browser:

- **Open Editor**

Die IDE verbindet sich auf:

- `ws://localhost:<ws-port>`
- Preview via `http://localhost:<astro-port>`

27.7 Nützliche Docker-Kommandos

Start

```
docker start phantomwp-my-site
```

Stop

```
docker stop phantomwp-my-site
```

Logs

```
docker logs -f phantomwp-my-site
```

Kompletter Reset

```
docker stop phantomwp-my-site
docker rm phantomwp-my-site
docker volume rm phantomwp-my-site
```

27.8 Mit lokalem Ordner statt Volume

Optional kannst du `/app` auf einen echten lokalen Ordner mounten, damit du mit anderen Tools direkt auf Dateien zugreifen kannst.

27.9 Healthcheck

```
curl http://localhost:14322/health
```

---

28. Deployment zu Vercel

Der empfohlene Standard-Host.

28.1 Vorteile

- Free Tier
- automatische Deployments
- HTTPS
- globales CDN
- Preview Deployments
- Custom Domains

28.2 Vercel verbinden

Token holen

1. bei Vercel einloggen
2. Settings
3. Tokens
4. Create Token
5. kopieren

In PhantomWP

1. Dashboard
2. Projekt finden
3. **Setup Deployment**
4. Token einfügen
5. Connect

28.3 Importieren

1. GitHub-Repo wählen
2. Team oder Personal Account
3. Einstellungen prüfen

4. **Deploy**

28.4 Standard Build Settings

- Build Command: `npm run build`
- Output Directory: `dist`
- Install Command: `npm install`

28.5 Deploy auslösen

Wichtig:
Deployment passiert nicht automatisch beim Editieren.

Du musst:

1. Änderungen machen
2. Git-Icon
3. Commit-Message
4. **Commit & Push**

Dann baut Vercel automatisch.

28.6 Custom Domain

In Vercel:

- Settings
- Domains
- Domain hinzufügen

DNS:

- Nameserver auf Vercel
- oder A/CNAME Records

SSL:

- automatisch

28.7 Rollback

In Vercel Dashboard:

- Deployments
- altes funktionierendes Deployment wählen
- **Promote to Production**

---

29. Deployment zu Cloudflare Workers

Alternative zu Vercel, besonders für server-mode Astro / API-Routes.

29.1 Voraussetzungen

Cloudflare API Token mit:

- Account Settings read
- Workers Scripts edit
- Workers KV Storage edit

29.2 Ablauf

1. Token erstellen
2. Dashboard öffnen
3. Cloudflare verbinden
4. ggf. Account ID eintragen
5. Projekt deployen

PhantomWP konfiguriert dabei:

- `@astrojs/cloudflare`
- `wrangler.jsonc`
- GitHub Actions Workflow
- GitHub Secrets für Deploy

29.3 Domains

Custom Domain aktuell über Cloudflare Dashboard verwalten.

---

30. WordPress Security / WordPress „offline“ nehmen

Extrem starkes Sicherheitsfeature.

30.1 Idee

Du blockierst öffentliche Zugriffe auf WordPress komplett,
erlaubst aber Requests mit dem Header:

`X-PhantomWP-Secret`

Dann kann:

- PhantomWP weiterhin Inhalte abrufen
- du mit Browser-Extension weiter ins WP-Admin
- Bots und Öffentlichkeit werden blockiert

30.2 Ablauf

Schritt 1

Secret Key in PhantomWP holen

Schritt 2

ModHeader im Browser installieren

Header setzen:

- Name: `X-PhantomWP-Secret`
- Value: dein Secret

Schritt 3

Auf Server Requests ohne Header blockieren

30.3 Optionen

Cloudflare Worker

empfohlen

Cloudflare Access

nginx

Apache `.htaccess`

30.4 Testen

Ohne Header:

```
curl -I https://your-wordpress-site.com
```

Erwartet:

- `403 Forbidden`

Mit Header:

```
curl -I -H "X-PhantomWP-Secret: YOUR_SECRET_KEY" https://your-wordpress-site.com
```

Erwartet:

- `200 OK`

30.5 Best Practices

- immer HTTPS
- WP trotzdem up-to-date halten
- starke Passwörter
- regelmäßige Backups
- Access Logs prüfen

---

31. Updating to RC.6

Falls du ein älteres PhantomWP-Projekt hast.

31.1 Wann relevant?

Nur für bestehende Projekte auf älteren Runtime-/Workspace-Versionen.

31.2 Codespace-Projekt updaten

1. Projekt öffnen
2. warten bis Codespace verbunden ist
3. Toolbar: **More → Infrastructure**
4. **Update**
5. warten
6. Tab reloaden

31.3 Docker-Projekt updaten

1. Projekt öffnen
2. Local run/start Dialog öffnen
3. generierten Update-Command kopieren
4. im Terminal ausführen
5. Container neu starten lassen
6. Editor neu öffnen

Wichtig:

- Projektdateien bleiben im Volume / lokalen Ordner erhalten

31.4 Wenn altes Projekt instabil ist

Empfehlung:

- neues RC.6-Projekt erstellen
- nur Site-Files kopieren:
  - pages
  - layouts
  - components
  - styles
  - public assets
  - content/data
  - bewusst geänderte Configs

Nicht kopieren:

- alte PhantomWP Runtime-/Workspace-Ordner

---

32. Troubleshooting – ultrakompakt

32.1 Codespace startet nicht

- Dashboard → Codespace entfernen
- neuen Codespace erstellen
- alternativ Docker lokal nutzen

32.2 Preview bleibt leer

- Codespace/Container läuft?
- 10–15 Sekunden warten
- Preview refreshen
- Services neu starten

32.3 Änderungen speichern nicht

- Status „Saved“ prüfen
- manuell `Cmd/Ctrl + S`
- unsaved dot auf Tab prüfen

32.4 WordPress verbindet nicht

- URL korrekt?

- `https://`
- keine `/wp-admin`
- REST API testen:
  - `https://yoursite.com/wp-json/wp/v2/posts`
- Security Plugins prüfen

32.5 AI funktioniert nicht

- API-Key eingetragen?
- Credits vorhanden?
- anderes Modell probieren
- Seite refreshen

32.6 Media Upload schlägt fehl

- Verbindung prüfen
- Dateityp prüfen
- Dateigröße prüfen
- ggf. einzeln hochladen

32.7 Menü erscheint nicht

- Menü gespeichert?
- Komponente importiert?
- Preview refreshed?

32.8 Component Library insert funktioniert nicht

- Datei geöffnet?
- geeigneter Dateityp?

- Cursor an sinnvoller Stelle?
- richtige Zieldatei im Header?

32.9 Vercel Deploy-Problem: „Login Connection“

Wenn Vercel meldet, dass eine GitHub Login Connection fehlt:

- in Vercel die **GitHub Authentication Connection** hinzufügen
- danach erneut deployen

32.10 Docker lokal wird nicht erkannt

- `docker ps`
- Healthcheck testen
- Browser-Erlaubnis für localhost geben
- Logs prüfen
- Token ggf. neu erzeugen

---

33. Die 10 wichtigsten Tastenkombinationen / Sofort-Merker

1. **Save** → `Cmd/Ctrl + S`
2. **Quick Open** → `Cmd/Ctrl + P`
3. **Find** → `Cmd/Ctrl + F`

4. **Replace** → `Cmd/Ctrl + H`
5. **Go to Line** → `Cmd/Ctrl + G`
6. **Format Document** → `Shift+Option+F` / `Shift+Alt+F`
7. **AI Modify** → `Cmd/Ctrl + Shift + M`
8. **AI Generate** → `Cmd/Ctrl + Shift + G`
9. **Inline AI Submit** → `Cmd/Ctrl + Enter`
10. **File Tree Search** → `/`

---

34. Die 15 wichtigsten Dateien/Ordner, die du kennen solltest

Pfad	Zweck
`src/pages/`	alle Seiten / Routen
`src/pages/index.astro`	Startseite
`src/pages/blog/`	Blogposts
`src/components/`	wiederverwendbare Komponenten
`src/components/menus/`	Menüs
`src/layouts/`	Layouts
`src/media/`	optimierte Bilder
`src/media/cms/`	heruntergeladene WordPress-Bilder
`public/`	statische Dateien
`src/lib/wordpress.ts`	WordPress-Client
`src/styles/theme.css`	Theme Studio Output
`src/config/menus.json`	Menü-Konfiguration
`src/data/products.json`	WooCommerce Produktdaten
`.env`	Verbindungs-/Secret-Konfiguration
`astro.config.mjs`	Astro-Konfiguration

---

35. Meine empfohlene Lernreihenfolge für PhantomWP

Wenn du schnell fit werden willst, arbeite in dieser Reihenfolge:

Phase 1: Grundlagen

1. Quick Start
2. IDE Overview
3. Working with Files
4. Component Library
5. Deploying to Vercel

Phase 2: Design-Workflow

6. Visual Editor
7. Media Manager
8. Theme Studio
9. Font Manager
10. Icon Manager
11. Menu Builder

Phase 3: AI-gestütztes Arbeiten

12. AI Assistant
13. Claude Code CLI

Phase 4: WordPress-Headless

14. Connecting WordPress
15. Browsing WordPress Data
16. Fetching Content
17. Image Optimization
18. WordPress Security

Phase 5: E-Commerce / Advanced

19. PhantomWP Connect Plugin
20. WooCommerce Setup
21. Store Features
22. Docker Local Development
23. Cloudflare Deployment

---

36. Mein empfohlenes „Sofort produktiv“-Setup

Wenn du **heute** damit starten willst, würde ich dir diesen Weg empfehlen:

Für einfache Unternehmens- oder Marketing-Site

1. GitHub verbinden
2. Projekt via Codespace erstellen
3. `index.astro` öffnen
4. Quick Start Template aus Component Library einfügen
5. Theme Studio anwenden
6. Menü im Menu Builder bauen
7. Bilder via Media Manager hochladen
8. Texte anpassen
9. Commit & Push
10. Vercel deployen

Für bestehende WordPress-Seite

1. WordPress verbinden
2. Content-Typen auswählen
3. Generate Pages
4. Media downloaden
5. Blog-/Page-Templates prüfen
6. Theme Studio anwenden
7. Navigation bauen
8. Deployen
9. optional: WordPress Security aktivieren

Für WooCommerce-Shop

1. WooCommerce Store Template wählen
2. PhantomWP Connect Plugin installieren
3. Pairing prüfen
4. WooCommerce verbinden
5. Produkte + Bilder syncen
6. BACS aktivieren für Tests
7. Cart/Checkout prüfen
8. Deployen
9. später Stripe/PayPal ergänzen

---

37. Ultrakurze „Merke dir das“-Zusammenfassung

Wenn du dir nur das Wichtigste merken willst:

- **PhantomWP = WordPress als CMS + Astro als statisches Frontend**
- **schnell, sicher, wenig Wartung**
- **Projekt läuft in Codespaces, Docker lokal oder Fly.io**
- **wichtigste Dateien:** `src/pages`, `src/components`, `src/layouts`, `src/lib/wordpress.ts`
- **WordPress Connect = read-only**
- **PhantomWP Connect Plugin = Full Access, Writes, Auth, Woo**
- **Component Library + Theme Studio = schnellster Weg zur fertigen Seite**
- **AI Assistant hilft beim Bauen, Refactoren, Debuggen**

- **Vercel = Standard-Deployment**
- **Commit & Push nötig, damit Deployment passiert**
- **WordPress optional komplett abschirmbar mit `X-PhantomWP-Secret`**

Revision #2
Created 2026-06-29 11:31:37 UTC by art10m
Updated 2026-06-29 12:01:46 UTC by art10m