

Der Block-Editor und die Zukunft von WordPress

Experten-Tipps für
Website-Betreiber



Der Redakteur

Oliver Lindberg

Oliver Lindberg ist ein unabhängiger Redakteur, Content-Consultant, und Gründer von Pixel Pioneers, einer Konferenz für Frontend-Entwickler und UX/UI Designer. Ehemals Chefredakteur der wegweisenden Zeitschrift 'net magazine', beschäftigt Oliver sich inzwischen seit mehr als 15 Jahren mit Webdesign und -entwicklung und hilft internationalen Unternehmen bei der Umsetzung von erfolgreichen Content(-Marketing)-Strategien.

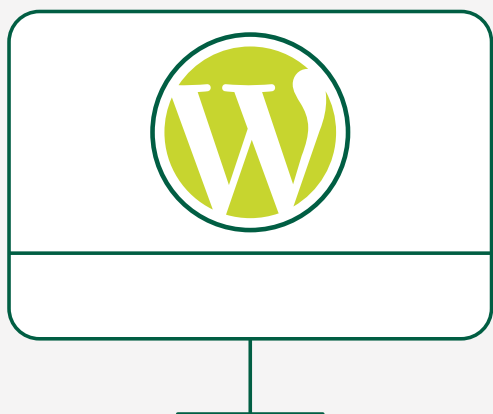
Vorwort

Mehr als ein Drittel aller Webseiten werden mit WordPress betrieben. Bei Webseiten, die ein Content Management System nutzen, liegt der Marktanteil sogar bei über 60 Prozent. Mit der Einführung des neuen Block-Editors Gutenberg hat nun ein neues Zeitalter für die Erstellung von Inhalten begonnen. Die Entwicklung bleibt weiterhin spannend.

In diesem E-Book möchten wir ein paar Anwendungsszenarien für die Arbeit mit dem Block-Editor vorstellen, Vorteile und Möglichkeiten besprechen, die Gutenberg jetzt schon mit sich bringt, und aufzeigen, wohin die Entwicklung geht.

Unsere Autoren sind alle Experten mit langjähriger WordPress-Erfahrung und aktive Mitglieder der deutschen WordPress-Community: Thordis Bonfranchi-Simović und [Vladimir Simović](#) – beide arbeiten seit über 17 Jahren mit WordPress – setzen sich in dem einleitenden Kapitel mit dem Pro und Contra von Gutenberg auseinander. Dann dringt [Kirsten Schelper](#), Designerin von WordPress-Webseiten mit Schwerpunkt Gutenberg und CSS sowie eine Hälfte des Blogs [Die Netzialisten](#), tiefer in zwei Konzepte ein, mit denen man im Block-Editor anspruchsvoll gestaltete Webseiten umsetzen kann: Block-Styles und Block-Patterns. Es folgt ein Ausblick auf künftige WordPress-Entwicklungen: [Matthias Kittsteiner](#) nimmt sich das Potenzial von Full-Site Editing vor, während [Ralf Wiechers](#) erklärt, wie Gutenberg in Kombination mit WooCommerce das Online-Einkaufserlebnis für Käufer wie auch Shop-Betreibende verbessern wird. Schließlich präsentiert Webentwickler [Simon Kraft](#), Mitglied des [Pluginkollektivs](#), eine Reihe von Tipps, wie man gute WordPress-Plugins erkennen und auswählen kann.

Viel Spaß beim Lesen!



Inhaltsverzeichnis

Gutenberg: Pro und Contra	6
Thordis Bonfranchi-Simović und Vladimir Simović	
Arbeiten mit Gutenberg: Block-Styles und Block-Patterns	20
Kirsten Schelper	
Full-Site Editing – ein Ausblick ins künftige WordPress	48
Matthias Kittsteiner	
Wie WooCommerce plus Gutenberg das Online-Einkaufserlebnis verändern wird	62
Ralf Wiechers	
Gute WordPress-Plugins erkennen und auswählen	74
Simon Kraft	



Die Autoren

Thordis Bonfranchi-Simović und Vladimir Simović

Thordis Bonfranchi-Simović arbeitet seit Januar 2004 mit WordPress.

Vladimir Simović arbeitet seit 2000 mit HTML und CSS und seit Januar 2004 mit WordPress. Im Laufe der Jahre hat er diverse Fachbücher und Fachartikel publiziert.

Die Eheleute sind Gesellschafter der Webagentur perun.net [webwork gmbh](http://webwork.gmbh), die umfangreiche Dienstleistungen rund um WordPress anbietet.

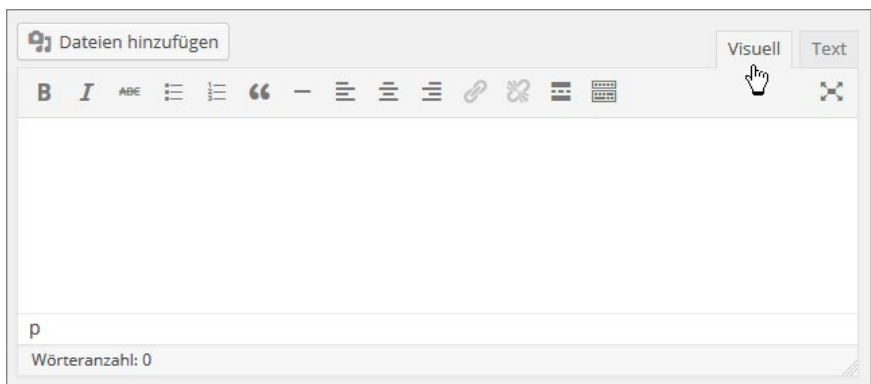
Gutenberg: Pro und Contra

[Gutenberg](#) ist den meisten von uns wohl als Erfinder des modernen Buchdrucks bekannt. Die Druckerpresse von Gutenberg hat eine Revolution in der Medienwelt und weitgehende Umwälzungen in der Gesellschaft angestoßen. Eine ähnlich starke Wirkung erhofften sich die Planer auch für die WordPress-Welt, und deswegen wurde das Projekt auf den Namen Gutenberg getauft.

Einführung

Als der Gutenberg-Editor mit der WordPress-Version 5.0 Anfang Dezember 2018 eingeführt wurde, war die Aufregung sehr groß. Die Debatten zum Für und Wider wurden hitzig geführt und werden es immer noch.

Zeit also, sich noch einmal in Erinnerung zu rufen, warum WordPress nicht mehr den alten TinyMCE nutzt, sondern nun auf den Gutenberg-Editor setzt.



Der alte TinyMCE-Editor ...



... und der neue Gutenberg-Editor.

Auf den ersten Blick erscheint der Gutenberg-Editor im Gegensatz zum TinyMCE recht spartanisch, aber das Gegenteil ist der Fall. Hinter dem Gutenberg-Editor, der nur den ersten Schritt einer Entwicklung darstellt, steckt die Absicht, WordPress zu einem Page-Builder-System „umzubauen“.

Um auch in Zukunft das führende CMS zu bleiben, war es für WordPress an der Zeit, sich neu zu erfinden, zumindest empfand dies [Matt Mullenweg](#) so.

Auf dem Theme-Markt waren und sind sehr häufig solche Themes kommerziell erfolgreich, die sogenannte Page-Builder mitliefern. Diese Themes versprechen ihren Nutzer*innen nicht nur, dass sie das grundsätzliche Design einer Website anpassen können, also so Dinge wie

Farben, Schriftarten, grundsätzliche Struktur (Sidebar links oder rechts, wie viele Spalten hat der Footer, Menüposition etc.), sondern auch, dass bei jedem Beitrag und bei jeder Seite jedes Element – also auch jeder Absatz – individuell gestaltet werden kann, ohne im Code einzugreifen und ohne CSS-Kenntnisse vorauszusetzen.

Die WordPress-Entwickler*innen haben sich schon immer die [Demokratisierung des Publizierens](#) auf die Fahne geschrieben, und mit dem Wechsel des Editors wollte man diese Demokratisierung weiter fördern.

Auch technisch wenig affine Nutzer*innen sollen mithilfe von Page-Buildern in der Lage sein, mit „wenigen“ Klicks eine ansprechende und funktionierende Website zu erstellen.

Diese neuen Möglichkeiten freuen einige, andere sehen darin aber auch eine Gefahr, und ob es dadurch wirklich einfacher ist, eine Website zu erstellen, bleibt dahingestellt. Neben der Möglichkeit, sich kreativ auszutoben, besteht nämlich durchaus auch die Gefahr, sich seine Website „kaputt zu spielen“.

Oft genug kommt es vor, dass Website-Betreiber*innen am Ende der kreativen Phase erschrocken feststellen, dass das Aussehen ihrer Website nun nichts mehr mit „Design“ zu tun hat. Dazu kommen nicht selten technische Probleme, die oft nur durch das komplette Neugestalten einer Website behoben werden können.

Auch nach erfolgreicher Erstellung und Gestaltung einer Website bleiben Page-Builder problemfällig. Eines der größten Probleme ist dabei der Theme-Wechsel.

Werden theme-eigene Elemente verwendet, die bei allen Page-Buildern im Code immer durch sogenannte Shortcodes eingefügt werden, werden diese von anderen Themes nicht erkannt und können nicht dargestellt werden. Die Shortcodes werden dann im Frontend sichtbar und müssen händisch entfernt bzw. umgewandelt werden.

Es war also nicht verwunderlich, dass sehr viele Theme-Entwickler*innen den Gutenberg-Editor nicht überschwänglich begrüßt haben und der Übergang doch recht holprig war.

Auch Plugin-Entwickler*innen waren nicht unbedingt begeistert. Viele Plugins sind mit ihren Funktionen Teil des Editors oder auf der Bearbeiten-Seite zusammen mit dem Editor integriert. All diese Plugins mussten umgestellt werden, um auch mit Gutenberg zu funktionieren. Noch heute werden allerdings Plugins von Benutzer*innen eingesetzt, die mit dem Gutenberg-Editor nicht kompatibel sind.

Gutenberg ausblenden

So war es nicht verwunderlich, dass zeitgleich mit Gutenberg das Plugin [Classic Editor](#) erschien. Dies ist ein offizielles Plugin, das vom WordPress-Team gepflegt wird und den früheren WordPress-Editor TinyMCE sowie die „Beitrag bearbeiten“-Oberfläche wiederherstellt. Es ermöglicht somit die Nutzung von Plugins, die diese Oberfläche erweitern oder auf andere Weise vom vorherigen Editor abhängen.

Der Classic Editor wird laut der Entwickler*innen bis mindestens 2022, oder so lange wie nötig, voll unterstützt und gewartet.

Mittlerweile gibt es eine Reihe von Plugins, die man unter dem Stichwort [disable Gutenberg](#) im Plugin-Verzeichnis findet.

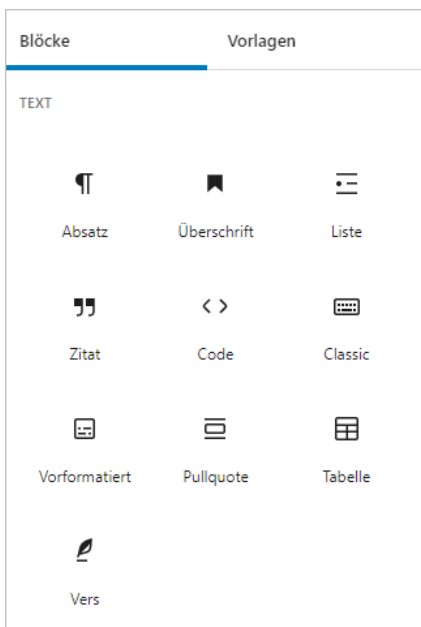
Wer damals ganz auf den Gutenberg Editor verzichten wollte, konnte sogar komplett aus WordPress „aussteigen“ und zu [ClassicPress](#) wechseln.

ClassicPress

Bei ClassicPress handelt es sich um ein CMS, das sich ausgehend von der WordPress-Version 4.9 entwickelt hat. Der alte TinyMCE Editor wird beibehalten. Die Entwickler*innen setzen dabei auf die folgenden Features: „Mächtig. Vielseitig. Vorhersehbar.“ Damit werden genau die Punkte angesprochen, die viele WordPress-Nutzer*innen beim Wechsel des Editors beklagt haben. Zum einen fiel dadurch eine gewisse Vielseitigkeit weg, da viele Plugins, die die Editor-Oberfläche beeinflusst haben, nun nicht mehr nutzbar waren, zum anderen fühlten sich viele Nutzer*innen, aber auch nicht wenige Entwickler*innen übergangen, und es wurde die Vorhersehbarkeit in der Entwicklung beklagt, da die überwiegende Mehrheit der WordPress-Community diese Entwicklung nicht hatte kommen sehen.

Gutenberg im Detail

Was genau hat Gutenberg nun gebracht? Das Projekt „Gutenberg in WordPress“ ist zu Beginn mit dem Einsatz des Editors auf den „Neu erstellen“ und „Bearbeiten“-Seiten für Beiträge und Seiten gestartet. Der Editor zeichnet sich dadurch aus, dass alle Elemente als Blöcke frei platziert und gestaltet werden können. Im Laufe der Zeit sind immer mehr Elemente hinzugekommen und auch die Möglichkeiten der Gestaltung sind erweitert worden.



Text-Blöcke im Gutenberg-Editor

Die Blöcke lassen sich in fünf Gruppen aufteilen:

1. Text
2. Medien
3. Design
4. Widgets
5. Einbettungen

Zu den Grundelementen bzw. den grundlegenden Blöcken aus den Bereichen Text und Medien gehören dabei unter anderem Überschriften, Textabsätze und Bilder.

Gutenberg bietet aber auch Design-Blöcke, um Beiträge bzw. Seiten zu gestalten wie zum Beispiel Abstandshalter oder Spalten. Zusätzlich dazu gibt es noch Widget-Blöcke, mit denen man automatisiert Inhalte einbinden kann. Dazu zählen beispielsweise die neuesten Kommentare, verschiedene Archive oder RSS Feeds. Die letzte Gruppe bilden Einbettungen, um Inhalte fremder Websites oder -dienste im eigenen Inhalt einzubetten.

Sowohl die Text- wie auch die Medien-Blöcke bieten den Nutzer*innen vielfältige Design-Möglichkeiten, die die Möglichkeiten des alten TinyMCE deutlich übersteigen.

Eine andere besondere Neuerung sind dabei die Widget-Blöcke. Diese Inhalte gab es bis dato bei WordPress standardmäßig nur für die Widgetbereiche (Sidebar, Footer etc.). Mit Gutenberg ist jetzt auch möglich, solche Elemente im Inhalt zu platzieren.

Somit bietet WordPress seinen Nutzer*innen mittlerweile einen Page-Builder für den Inhaltsbereich, den man frei mit verschiedenen Elementen und Blöcken gestalten kann.

Konkret bedeutet dies, dass die Trennung von Design und Inhalt verschwimmt. Der Leitsatz, der bis vor kurzem im Web noch galt, diese Elemente nicht zu vermischen, scheint für WordPress nicht mehr zu gelten. Er wurde aufgegeben. Diese Entwicklung ist grundsätzlich im Web zu beobachten und hängt eng mit der Entwicklung der Page-Builder, also Baukasten-Websites zusammen.

Das Ziel von Gutenberg soll sein, dass die komplette Website, also auch der Header und Footer sowie Sidebareaeiche, mit der Hilfe des Gutenberg-

Editors gestaltet werden können. Die Entwicklung hin zu diesem „Full-Site-Editing“ ist bis jetzt allerdings nur zu erahnen.

Die Entwicklung hat sich bisher schwieriger gestaltet als man sich das vor zwei Jahren vorgestellt hatte. Viele Features wurden erst einmal auf Eis gelegt oder verschoben. Dazu zählen unter anderem die Möglichkeit, den Widgetbereich sowie das Menü mithilfe von Gutenberg zu gestalten.

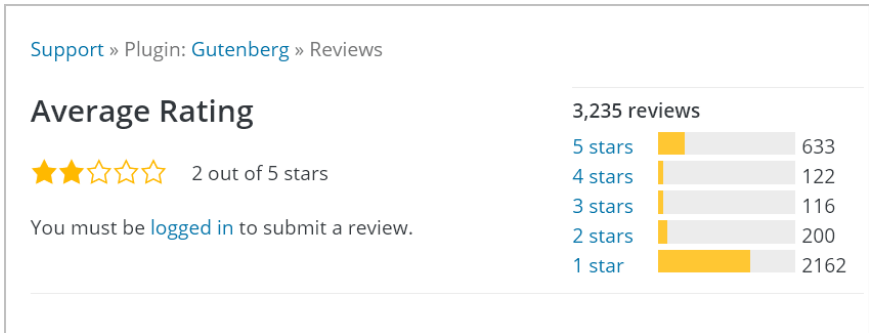
Wer die aktuelle Entwicklung ein wenig im Voraus kennenlernen möchte, kann sich dazu das offizielle [Gutenberg-Plugin](#) installieren. Hierdurch erhält man schon Features, die erst in zukünftigen WordPress-Versionen Teil des Core-Programms sein werden.

Die Bewertung des Plugins auf der offiziellen Seite lässt erahnen, dass Gutenberg auch heute noch äußerst kritisch gesehen wird.

Kritik an Gutenberg

Wer Gutenberg und seine Entwicklung kritisiert, der riskiert leider recht häufig als Fortschrittsverweigerer oder Ähnliches klassifiziert zu werden. Wie in anderen Bereichen des Webs ist es leider auch in der WordPress-Welt nicht möglich, ohne Grabenkämpfe auszukommen.

Es mag in der Tat unter Gutenberg-Skeptikern auch welche geben, die mit jeglicher Neuerung nicht gut zurechtkommen oder nicht zurechtkommen wollen, aber diese Gruppe dürfte die absolute Minderheit darstellen.



Screenshot von dem Bewertungsstatus des Gutenberg-Plugins.

Wer einen Blick auf die Bewertungen des Gutenberg-Plugins wirft, wird sehr schnell feststellen, dass sich der Unmut vieler WordPress-Nutzer*innen in den schlechten Bewertungen wiederfindet. Da die Nutzer*innen keine direkten Möglichkeiten haben, sich über die Entwicklung von WordPress selber zu beschweren, entlädt sich wohl viel Kritik hier beim Plugin. Ernst nehmen sollte man sie trotzdem.

Natürlich können wir gerne diskutieren, ob so eine Gesamtbewertung wirklich gerechtfertigt sei, aber wir alle können uns darauf einigen, dass Gutenberg in großen Teilen der WordPress-Community sehr umstritten ist.

Wo liegen die Probleme?

WordPress ist ein etabliertes und in der Praxis erprobtes System, welches seit Mitte 2003 existiert. Dementsprechend gibt es viele Menschen, Organisationen und Firmen, die das CMS seit vielen Jahren erfolgreich nutzen und ihre eigenen Arbeitsabläufe eingerichtet haben.

Dass sich hierbei die Freude auf einen Umstieg in Grenzen hält, ist nicht verwunderlich. Daher ist es auch nicht überraschend, dass sich die beiden Plugins [Classic Editor](#) und [Disable Gutenberg](#) großer Beliebtheit erfreuen.

Das erste Plugin ist auf mindestens fünf Millionen WP-Installationen aktiv und reaktiviert den klassischen Editor, sodass die Möglichkeit besteht, sowohl mit Gutenberg als auch mit der klassischen Oberfläche zu arbeiten.

Disable Gutenberg ist auf mindestens 600.000 WP-Installationen aktiv und deaktiviert Gutenberg nahezu komplett.

Allerdings gibt es auch jenseits des Arguments „wir haben etwas Etabliertes, wir brauchen nichts Neues“ einige Aspekte, die vielen Nutzer*innen nicht zusagen.

In der täglichen Arbeit [manifestieren sich](#) immer noch einige Unzulänglichkeiten von Gutenberg. Speziell bei längeren Texten fallen die vermeintlich kleinen Nachteile schnell auf.

Ab einer gewissen Länge des Textes macht sich bemerkbar, dass Gutenberg doch um einiges leistungshungriger ist als der alte TinyMCE Editor.

Das macht sich entweder mit einem lauterem Lüfter am Laptop oder bei passiv gekühlten Systemen in einer Verringerung der Systemleistung bemerkbar.

Die Tatsache, dass Gutenberg den Text, oder besser gesagt, den Inhalt eines Blogartikels oder einer Seite in einzelne unabhängige Blöcke

separiert, widerspricht dem Empfinden vieler Autor*innen, die den Inhalt des Artikels als eine Einheit empfinden.

Aber jenseits dieser Empfindung entstehen dadurch auch handfeste Nachteile, zum Beispiel bei der Arbeit mit Rechtschreibtools.

Wir nutzen sehr gerne das browserbasierte Rechtschreib- und Grammatik-Tool von [LanguageTool](#). Dieses Werkzeug prüft die Inhalte von Textboxen auf korrekte Rechtschreibung und Grammatik.

Der komplette Inhalt eines Blogartikels oder einer Seite war bei dem alten Editor in einer einzigen Textbox, bei Gutenberg dagegen ist jeder einzelne Block eine unabhängige Textbox. Das heißt, dass für die Textprüfung jede Box einzeln angesteuert werden muss. Das dürfte auch bei anderen Tools, die Textboxen ansteuern, nicht anders sein.

Nach unserer Einschätzung ist die Anforderung an die Personen, die in unregelmäßigen Zeitabständen die Inhalte der Website pflegen müssen oder sollen, gestiegen. Der alte, oder besser gesagt, der klassische Editor hat in seinem Aufbau an die Erscheinung erinnert, die viele Nutzer*innen von Textverarbeitungen wie Word oder LibreOffice kannten.

Die Textansicht des alten Editors erinnert an viele Kontaktformulare oder die Editoren, wie viele Leute sie auch aus den Foren kennen. Also etwas, was den Nutzer*innen recht häufig im Web begegnet.

Damit war die Einstiegshürde im Fall des alten Editors für unbedarfte Nutzer*innen vergleichsweise gering.

Der neue Editor schaut nicht nur anders aus, sondern ist schon jetzt leistungsfähiger. Eine höhere Leistungsfähigkeit geht allerdings logischerweise immer auch mit einem Mehr an Funktionen daher, in die sich neue Nutzer*innen noch einarbeiten müssen. Darum ist die Hürde für Menschen, die „schnell mal etwas einfügen wollen“, merklich gestiegen.

Die gestiegene Hürde ist nicht nur an sich ein Problem, sondern führt zu einer weiteren problematischen Einschätzung unsererseits.

Der kometenhafte Aufstieg von WordPress war unter anderem der Tatsache geschuldet, dass die Installation, die Einrichtung, der Betrieb, aber auch die Erstellung neuer Inhalte vergleichsweise einfach waren.

Diese Einfachheit – im besten Sinne des Wortes – hat nicht nur zur Popularität von WordPress geführt, sondern hat auch das Bloggen aus dem absoluten Nischendasein befreit und somit maßgeblich zu der Vielfalt und zu der Demokratisierung des Publizierens im Web beigetragen.

Natürlich ist WordPress nach wie vor zurecht das mit Abstand [populärste CMS](#). Aber wer lange genug im Web unterwegs ist, wird mindestens eine Handvoll Beispiele von ehemals sehr populären Projekten kennen, die mittlerweile in der Versenkung verschwunden sind.

Fazit nach zwei Jahren Gutenberg

Ziemlich genau zwei Jahre nach der Veröffentlichung von Gutenberg kann man sagen, dass sich die Aufregung zwar gelegt hat, aber die Kritik an Gutenberg geblieben ist. Auch die Tatsache, dass die Entwicklung langsamer als geplant vorangeht und die Befürworter von Gutenberg schon deutlich früher mit einem Full-Site-Editor gerechnet haben, hat den anfänglichen Enthusiasmus über WordPress mit den Möglichkeiten eines Page-Builders einen deutlichen Dämpfer verpasst.

Eines kann man aber gewiss sagen, die Entwicklung von WordPress bleibt spannend! WordPress wird sich zu einem Full-Site-Editing-CMS entwickeln und dadurch neue Zielgruppen ansprechen. WordPress wird eine ernste Konkurrenz zu allen bisherigen Page-Buildern, die dann nicht mehr ergänzend als Plugin installiert werden müssen.

Aber auch der Plugin-Markt selber wird sich deutlich verändern und erweitern. Schon jetzt können Blöcke wie Plugins installiert werden, um die Möglichkeiten der Gestaltung und Einbindung von Inhalten zu erweitern. Auch Theme-Entwickler*innen werden hier sicherlich Vorlagen entwickeln, die dann via Plugin eine Standard-WP-Installation ergänzen und den Nutzer*innen so mehr Möglichkeiten in der Gestaltung ihrer Website bieten.



Die Autorin

Kirsten Schelper

Kirsten Schelper ist studierte Kommunikationsdesignerin. Sie hat sich früh auf das Web spezialisiert und arbeitet seit über 10 Jahren mit WordPress. Schwerpunkte ihrer Arbeit sind Websites mit dem neuen Gutenberg-Editor. Sie schreibt im Blog [Die Netzialisten](#) über CSS, WordPress und Gutenberg und ist Mitbegründerin des [WordPress Meetups München](#).

Arbeiten mit Gutenberg: Block-Styles und Block-Patterns

In diesem Artikel möchte ich zwei Konzepte vorstellen, über die Sie mit dem Gutenberg-Editor schöne, anspruchsvolle Designs umsetzen können. Über **Block-Styles** können Sie einem Gutenberg-Block ein individuelles Design mitgeben. Mit **Block-Patterns** stellen Sie in Ihrem Projekt wiederverwendbare Layoutvorlagen zur Verfügung. Wenn Sie gute Grundkenntnisse in CSS und HTML haben, können Sie Block-Styles und Block-Patterns selbst in Ihr Theme einbauen. Alternativ können Sie sich ein Theme suchen, in das die Autor*innen Block-Styles und Block-Patterns bereits integriert haben.

Block-Styles

Gutenberg bringt einige Block-Styles schon mit. So gibt es z. B. für den Zitat-Block zwei unterschiedliche Designs. Links ist ein Zitat im Standard-Design zu sehen, rechts eine alternative Variante mit größerem Text:



Auch den Button-Block gibt es in zwei Versionen, einmal mit Hintergrundfarbe und einmal mit transparentem Hintergrund:



Alle Block-Styles werden in der rechten Seitenleiste angezeigt. Für jeden Block-Style wird automatisch ein kleines „Thumbnail“ generiert, so dass man das Aussehen des Designs gut erkennen kann.

Ein Gutenberg-Block kann mehrere Designvarianten haben. Über das Dropdown-Feld „Standard-Stil“ kann man festlegen, welches Design beim Einfügen des Blocks standardmäßig ausgewählt werden soll.

Was ist der Unterschied zwischen Block-Styles und Custom CSS?

Eine Designvariante für einen Block könnte man auch über eine zusätzliche CSS-Klasse umsetzen. Diese Klasse trägt man unter „Erweitert/Zusätzliche CSS-Klassen“ ein.

Erweitert ^

HTML-Anker

Gib ein oder zwei Wörter - ohne Leerzeichen - ein, um eine eindeutige Webadresse nur für diese Überschrift zu erstellen, die als "Anker" bezeichnet wird. Dann kannst du direkt zu diesem Abschnitt deiner Seite verlinken.
[Mehr über Anker erfahren](#) 

Zusätzliche CSS-Klasse(n)

Trenne mehrere Klassen durch Leerzeichen.

Dazu muss die CSS-Klasse natürlich im Theme-CSS (style.css) vorhanden sein.

Die Arbeit mit CSS-Klassen empfinde ich persönlich als wenig intuitiv. Das Feld für die zusätzlichen Klassen liegt etwas versteckt.

Wenn man nach einer Weile, in der man nicht mehr am Projekt gearbeitet hat, den Faden wieder aufnimmt, übersieht man leicht, dass eine CSS-Klasse im Einsatz ist. Außerdem sieht man den Effekt, den das CSS hat,

nur im Frontend. Wenn man möchte, dass das Design auch im Backend angezeigt wird, muss man zusätzlich zum Theme-CSS auch die Editor-Styles ergänzen.

Block-Styles sind hier intuitiver und einfacher in der Handhabung. Man sieht auf den ersten Blick, welche Varianten es gibt. Niemand muss sich die Namen von CSS-Klassen merken, und die Entwickler*innen müssen nicht eigens dafür sorgen, dass das CSS sowohl im Backend als auch im Frontend auftaucht. Das passiert bei den Block-Styles automatisch.

Beispiel 1: Absatz als Box

Technische Anmerkung: Meistens werden Block-Styles in ein Plugin ausgelagert. Diesen Ansatz verfolge ich auch hier. Auf diese Weise kann man Block-Styles über mehrere Projekte hinweg verwenden.

Ich möchte zum normalen Absatz-Block eine Designvariante mit einem roten Hintergrund hinzufügen. Der Absatz-Block soll so aussehen:

Das ist ein Absatz mit Block Styles.

Sed posuere consectetur est at lobortis. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.

Die Datei-Struktur

Für die Block-Styles brauchen wir drei Dateien:

- eine PHP-Datei (index.php)
- eine CSS-Datei (block-styles.css)
- eine JavaScript-Datei (blocks.js)

```
|— ks-block-styles
    |— blocks.js
    |— block-styles.css
    |— index.php
```

Das CSS

Als erstes überlege ich mir mein CSS. Die Box soll eine rote Hintergrund-Farbe bekommen. Den Text möchte ich zentriert ausrichten.

Das würde so aussehen:

```
.paragraph-red-box {
  padding: 1em;
  background-color: #dc143c;
  color:white;
  text-align: center;
  border-radius:6px;
}
```

Für die bessere Lesbarkeit verwende ich im Namen der CSS-Klasse den Namen des Blocks (paragraph) und kombiniere ihn mit einem Stichwort zur Designvariante (box). Dieses CSS speichere ich in einer Datei namens block-styles.css.

Das JavaScript

Als nächstes brauche ich eine JavaScript-Datei. Da steht nicht viel drin außer dem Namen der CSS-Klasse `paragraph-box` und einer kurzen Beschreibung (**Rote Box**). Und natürlich findet sich dort auch die Bezeichnung des Blocks, auf den die Block-Styles angewendet werden sollen. In unserem Fall ist das der Absatz-Block `core/paragraph`.

```
wp.blocks.registerBlockStyle( 'core/paragraph', {  
  name: 'paragraph-red-box',  
  label: 'Rote Box'  
} );
```

Die PHP-Datei

Es geht weiter mit der `index.php`. Das ist unsere Haupt-Plugin-Datei. Sie enthält auch den Plugin-Header, damit unser Block-Styles-Plugin als Plugin erkannt wird.

In dieser Datei binde ich die JavaScript-Datei und die CSS-Datei über die Hooks `enqueue_block_assets` und `enqueue_block_editor_assets` ein.

```

1  <?php
2
3  /**
4   * Plugin Name: Kirstens Gutenberg Block Styles
5   * Plugin URI: https://die-netzialisten.de
6   * Description: Block Styles für Absatz-Block und Listen-Block
7   * Version: 1.0
8   * Author: Kirsten Schelper
9   */
10
11 /**
12  * Register Custom Block Styles
13  */
14
15 if ( ! defined( 'ABSPATH' ) ) exit;
16 /**
17  * Enqueue Block Styles Javascript
18  */
19 function kirstens_gutenberg_scripts() {
20     wp_enqueue_script(
21         'kirstens-block-styles-js',plugins_url( '/js/blocks.js', __FILE__ ),
22         array( 'wp-blocks', 'wp-dom-ready', 'wp-edit-post' ),filemtime( plugin_dir_path( __FILE__ ) . '/js/blocks.js' )
23     );
24 }
25 add_action( 'enqueue_block_editor_assets', 'kirstens_gutenberg_scripts' );
26
27
28 /**
29  * Enqueue Block Styles Stylesheet
30  */
31 function kirstens_gutenberg_styles() {
32     wp_enqueue_style( 'kirstens-block-styles-css',plugins_url( '/css/block-styles.css', __FILE__ )
33     );
34 }
35 add_action( 'enqueue_block_assets', 'kirstens_gutenberg_styles' );

```

Zum Code →

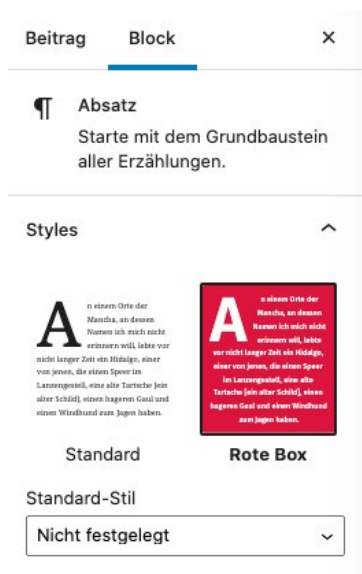
Eine Sache noch: Wir müssen eine kleine Änderung in der CSS-Datei machen. Damit das CSS von WordPress korrekt gelesen wird, muss man vor den Namen der CSS-Klasse ein `is-style` setzen.

Das fertige CSS sieht also so aus:

```
.is-style-paragraph-red-box {  
padding: 1em;  
background-color: #dc143c;  
color:white;  
text-align: center;  
border-radius:6px;  
}
```

Block-Styles anwenden

Wenn ich jetzt in den Editor gehe und den Absatz-Block wähle, taucht in der rechten Seitenleiste unter „Block/Styles“ ein zusätzliches Design auf, das ich auswählen kann.



Das neue Absatz-Design erscheint im Editor unter „Block/Styles“.

Beispiel 2: Liste mit grünen Häkchen

Wenn die Grundstruktur von meinen Block-Styles steht, kann ich ganz leicht weitere Block-Styles hinzufügen.

Dazu muss ich lediglich die CSS-Datei und die JavaScript-Datei um die Anweisungen für die zusätzlichen Styles ergänzen. An der PHP-Datei muss ich nichts verändern.

Angenommen, ich möchte eine Liste haben, die so aussieht:

- ✓ Listenpunkt 1
- ✓ Listenpunkt 2
- ✓ Listenpunkt 3

Dann ergänze ich mein JavaScript und CSS um die entsprechenden Anweisungen.

JavaScript:

```
//List with checkmarks
wp.blocks.registerBlockStyle( 'core/list', {
  name: 'checkmark-list',
  label: 'Liste mit Häkchen',
} );
```

CSS:

```
/*List with checkmark */  
.is-style-checkmark-list {  
  list-style: none !important;  
  padding-left: 0;  
}
```

```
.is-style-checkmark-list li:before {  
  content: '✓';  
  padding-right: 0.5em;  
}
```

Die Block-Styles für die Liste sind jetzt in der Sidebar des Editors sichtbar.



Das neue Listen-Design erscheint im Editor unter „Block/Styles“.

Hier der vollständige Code der beiden Block-Styles als Plugin:

```

1  //Paragraph Box
2
3  wp.blocks.registerBlockStyle( 'core/paragraph', {
4      name: 'paragraph-red-box',
5      label: 'Rote Box'
6  } );
7
8
9  //List with checkmarks
10
11 wp.blocks.registerBlockStyle( 'core/list', {
12     name: 'checkmark-list',
13     label: 'Liste mit Häkchen',
14 } );

```

[Zum Code →](#)

block-styles.js

```

1  /*Paragraph Red Box*/
2
3  .is-style-paragraph-red-box {
4      padding: 1em;
5      background-color: #dc143c;
6      color:white;
7      font-family: "fira_sansmedium";
8      text-align: center;
9      border-radius:6px;
10 }
11
12 /*List with checkmark */
13 .is-style-checkmark-list {
14     list-style: none !important;
15     padding-left: 0;
16 }
17
18 .is-style-checkmark-list li:before {
19     content: '✓';
20     padding-right: 0.5em;
21 }
22 }

```

[Zum Code →](#)

block-styles.css

```
1  <?php
2
3  /**
4   * Plugin Name: Kirstens Gutenberg Block Styles
5   * Plugin URI: https://die-netzialisten.de
6   * Description: Block Styles für Absatz-Block und Listen-Block
7   * Version: 1.0
8   * Author: Kirsten Schelper
9   */
10
11 /**
12  * Register Custom Block Styles
13  */
14
15 if ( ! defined( 'ABSPATH' ) ) exit;
16 /**
17  * Enqueue Block Styles Javascript
18  */
19 function kirstens_gutenberg_scripts() {
20     wp_enqueue_script(
21         'kirstens-block-styles-js',plugins_url( '/js/blocks.js', __FILE__ ),
22         array( 'wp-blocks', 'wp-dom-ready', 'wp-edit-post' ),filetime( plugin_dir_path( __FILE__ ) . '/js/blocks.js' )
23     );
24 }
25 add_action( 'enqueue_block_editor_assets', 'kirstens_gutenberg_scripts' );
26
27
28 /**
29  * Enqueue Block Styles Stylesheet
30  */
31 function kirstens_gutenberg_styles() {
32     wp_enqueue_style( 'kirstens-block-styles-css',plugins_url( '/css/block-styles.css', __FILE__ )
33     );
34 }
35 add_action( 'enqueue_block_assets', 'kirstens_gutenberg_styles' );
```

[Zum Code →](#)

[index.php](#)

Wo kommen Block-Styles zum Einsatz?

Immer dann, wenn man eine Variante eines Gutenberg-Blocks braucht, die sich gut per CSS umsetzen lässt, sind Block-Styles die Methode der Wahl. Mit Block-Styles können Sie das Aussehen von jedem Gutenberg-Block individualisieren.

Block-Styles sind gut für Websites geeignet, an denen Redakteure arbeiten, die einfach nur den Gutenberg-Editor anwenden möchten.

Block-Styles beziehen sich immer nur auf einen einzelnen Block. Möchte man mehrere Blöcke kombinieren und dieser Kombination ein individuelles Aussehen geben, dann kommt eine Block-Pattern in Frage.

Block-Patterns

Block-Patterns sind Layout-Vorlagen, wie man sie aus der Welt der Page-Builder kennt. In den Vorlagen werden mehrere Blöcke miteinander kombiniert. Mit diesen Vorlagen lassen sich anhand von wenigen Klicks anspruchsvolle Layouts aufbauen.

Ich möchte zwei Beispiele zeigen: einmal ein Testimonials-Modul und einmal eine Preistabelle. Für jedes Modul werde ich jeweils eine Block-Vorlage (die deutsche Übersetzung für Block-Pattern) erstellen.

Beispiel 1: Testimonials

Was unsere Kunden sagen

*Lorem ipsum dolor sit
amet, consectetur
adipiscing elit. Aenean
commodo ligula eget
dolor. Aenean massa.*

—Vorname Name, Firma

*Lorem ipsum dolor sit
amet, consectetur
adipiscing elit. Aenean
commodo ligula eget
dolor. Aenean massa.*

—Vorname Name, Firma

*Lorem ipsum dolor sit
amet, consectetur
adipiscing elit. Aenean
commodo ligula eget
dolor. Aenean massa.*

—Vorname Name, Firma

Ich verwende für diese Block-Pattern eine Kombination von Standard-Modulen. Ich arbeite ohne zusätzliches CSS.

Im ersten Schritt baue ich meine Testimonial-Modul ganz normal im Gutenberg-Editor auf. Ich füge eine Überschrift ein, lege drei Spalten an und setze in jede Spalte einen Zitat-Block. Abschließend gruppiere ich das Modul. Der Gruppe gebe ich eine CSS-Klasse mit, für den Fall, dass ich das Modul doch einmal per CSS ansprechen möchte.

Schritt 1: Block-Vorlage per Funktion registrieren

Als nächstes registriere ich die neue Block-Vorlage per Funktion. Das kann ich in der `functions.php` direkt im Theme machen. Ich könnte auch ein Plugin für die Block-Patterns schreiben. Dann sind die Patterns für mehrere Projekte nutzbar.

Die Funktion, mit der die Block-Pattern registriert wird, heißt `register_block_pattern`. Jede Block-Pattern hat einen Titel (`title`), eine Beschreibung (`description`), und man kann die Kategorie (`category`) definieren, unter der die Vorlage erscheinen soll. Und einen Inhalt (`content`) hat die Vorlage natürlich auch.

```

register_block_pattern(
    'beispiel',

    array(
        'title'           => The visible name in the editor,
        'viewportWidth' => The width of the pattern preview
    (int),
        'categories'      => An array of categories,
        'description'     => A description of the pattern,
        'keywords'        => An array of keywords used in the
search,
        'content'         => The block comment and markup,
    )

);

```

Beispiel aus <https://fullsiteediting.com/lessons/introduction-to-block-patterns/>.

Schritt 2: Block-Code kopieren

Jetzt brauchen wir noch den Code für die Vorlage. Dazu gehe ich an die Stelle, wo ich die Block-Vorlage erstellt habe (ein Beitrag oder eine Seite) und schaue mir den Beitrag in der Code-Editor-Ansicht von Gutenberg an. Diese Ansicht erreichen Sie über die drei Pünktchen ganz rechts oben.

Jetzt noch die Block-Pattern suchen und den entsprechenden Bereich in die Zwischenablage kopieren.

Schritt 3: Code in die Funktion kopieren

Den Code-Schnipsel füge ich jetzt an der Stelle in der `register_block_pattern`-Funktion ein, wo `content` steht.

Damit ist die Block-Vorlage fertig. Sie erscheint jetzt im Editor unter „Vorlagen“.

Der Code für meine Testimonials-Vorlage sieht im Ganzen so aus:

```
1 register_block_pattern(  
2 'ks-card-pattern',  
3 array(  
4 'title' => __( 'Beschreibung Vorlage', 'ks-block-pattern' ),  
5 'description' => _x( 'Beschreibung Vorlage', 'Beschreibung Vorlage', 'ks-block-pattern' ),  
6 'categories' => array('columns'),  
7 'content' => 'Hier kommt der Code für die Vorlage hin',  
8 )  
9 );
```

[Zum Code →](#)

Beispiel 2: Preistabelle

BASIC 9,99€ pro Monat Hier stehen die Features, für dieses Preismodell. Jetzt kaufen	STANDARD 14,99€ pro Monat Hier stehen die Features, für dieses Preismodell. Jetzt kaufen	PREMIUM 19,99€ pro Monat Hier stehen die Features, für dieses Preismodell. Jetzt kaufen
--	--	---

Bei dieser Vorlage setze ich zusätzliches CSS ein. Dieses CSS muss im Theme vorhanden sein. Wenn ich also die Block-Vorlage per Plugin einbauen würde, müsste ich dem Plugin eine CSS-Datei mit den notwendigen Styles mitgeben.

Registrierte ich die Vorlage direkt im Theme, muss das CSS im Theme-CSS (style.css) und in den Editor-Styles enthalten sein. Fehlt das CSS in den Editor-Styles, wird die Vorschau im Inserter nicht richtig angezeigt.

Grundlage der Pattern ist wieder der Spalten-Block mit drei Spalten. In jede Spalte kommen fünf Blöcke: Die erste Zeile ist eine Überschrift, darauf folgen drei Absatz-Blöcke und ein Button.

Ich gruppriere sowohl die drei inneren Blöcke als auch das gesamte Module. So kann ich jeweils eine CSS-Klasse vergeben, über die ich die Elemente gut ansprechen kann.

Das CSS:

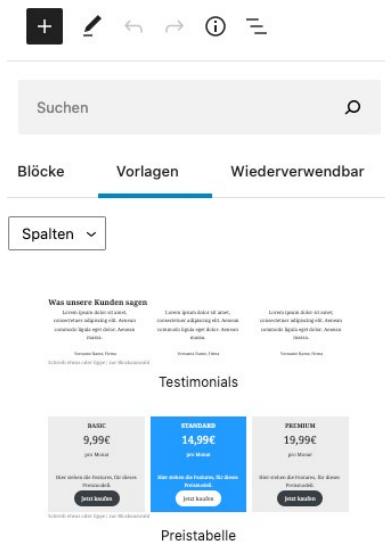
```
1  /*Resets*/
2  .preistabelle .wp-block-column:not(:first-child) {
3      margin-left: 1em;
4  }
5  .preistabelle p{
6      margin-top:0;
7  }
8
9  .preistabelle .wp-block-group{
10     padding:1em 0.5em;
11 }
12
13 .preistabelle .has-large-font-size{
14     font-size:50px;
15     margin:0
16 }
17
18 /*Button zentrieren*/
19
20 .preistabelle .wp-block-button{
21     margin:0 auto !important;
22 }
23 /*Hintergrundfarben*/
24
25 .preistabelle .basic{
26     background-color:#eaeaea;
27 }
28 .preistabelle .standard{
29     color:white;
30     background-color:DodgerBlue;
31 }
32 .preistabelle .premium{
33     background-color:#eaeaea;
34 }
```

[Zum Code →](#)

Hier der Code für die beiden Block-Patterns, Testimonials und Preistabelle:

[Zum Code →](#)

Beide Vorlagen erscheinen jetzt in der Auswahl für die Vorlagen.



Core-Patterns deaktivieren

In der Regel braucht man die Core-Patterns nicht, die Gutenberg standardmäßig mitbringt. So können Sie sie deaktivieren:

```
function removeCorePatterns() {
    remove_theme_support('core-block-patterns');
}
add_action('after_setup_theme', 'removeCorePatterns');
```

Patterns-Kategorien deaktivieren und eigene Kategorien erstellen

Auch die Standard-Kategorien kann man deaktivieren, wenn man sie nicht nutzen möchte.

```
1
2 function removeCorePatternsCategories() {
3     unregister_block_pattern_category('buttons');
4     unregister_block_pattern_category('columns');
5     unregister_block_pattern_category('gallery');
6     unregister_block_pattern_category('header');
7     unregister_block_pattern_category('text');
8     unregister_block_pattern_category('uncategorized');
9 }
10 add_action('init', 'removeCorePatternsCategories');
```

[Zum Code →](#)

So erstellen Sie eine eigene Kategorie für Ihre Block-Patterns:

```
1 function netzialisten_register_block_categories() {
2     if ( class_exists( 'WP_Block_Patterns_Registry' ) ) {
3         register_block_pattern_category(
4             'Netzialisten',
5             array( 'label' => _x( 'Patterns Netzialisten', 'Block pattern category', 'netzialisten' ) )
6         );
7     }
8 }
9 add_action( 'init', 'netzialisten_register_block_categories' );
```

[Zum Code →](#)

Wo kommt die Funktion für die Block-Patterns hin?

Den Code zu Block-Patterns können Sie direkt in die functions.php von Ihrem [Child-Theme](#) einfügen. Das ist aber ein bisschen unpraktisch, weil die functions.php dadurch recht unübersichtlich wird.

Es bietet sich daher an, die Block-Patterns in ein eigenes Funktions-Modul auszulagern. Dieses Modul kann man dann per **require** in die functions.php des Themes einbinden.

```
require get_template_directory() . '/inc/block-patterns.php';
```

Eine weitere Option wäre, für die Organisation der Block-Patterns ein Plugin zu nutzen, das solche und ähnliche Code-Schnipsel verwaltet. Meine Empfehlung dafür wäre das Plugin [Code Snippets](#).

Der Unterschied zwischen Block-Patterns und wiederverwendbaren Blöcken

Wenn man eine Block-Pattern wählt und in eine Seite einfügt, wird die Pattern damit Teil der Seite. Die Block-Pattern ist nach dem Einfügen nicht mehr mit der Vorlage verknüpft. Würde irgendjemand die Vorlage nachträglich ändern, würde man davon auf den Seiten nichts merken.

Bei wiederverwendbaren Blöcken ist das anders. Diese Blöcke sind für Elemente gedacht, die mehrfach auf Ihrer Website vorkommen und überall identisch ausschaun und denselben Inhalt haben.

Ein Beispiel wäre ein Block mit Ihrer Telefonnummer. Der Block sieht überall gleich aus und hat denselben Inhalt. In Wirklichkeit handelt es sich

hier um immer denselben Block. Er ist nur mehrfach verknüpft. Ändern Sie die Telefonnummer, dann ändert sich die Nummer in allen vorkommenden Verknüpfungen.

Block-Patterns – Bedeutung und Ausblick

Ich habe von Anfang an gern mit Gutenberg gearbeitet. Kam ich einmal mit den Standard-Einstellungen eines Blocks nicht weiter, habe ich mir schnell ein paar Zeilen CSS geschrieben. So konnte ich eigentlich so gut wie alles mit Gutenberg umsetzen, was ich gerne machen wollte.

Viele Anwender*innen erleben das anders. Sie kommen ein Stück weit voran – und bleiben dann stecken. Weil sie einen Abstand nicht hinbekommen oder eine Farbe nicht so vergeben können, wie sie möchten. Viele geben an dieser Stelle frustriert auf und gehen zurück zu ihrem gewohnten Page-Builder-Theme.

Page-Builder haben für jedes Modul Einstellungsoptionen für alle Details. Da ist Gutenberg (noch) nicht. Es fehlen nach wie vor die kleinen Einstellungsoptionen bei den Blöcken. Aber die Optionen werden mehr. Mit jedem Release gibt es Verbesserungen und Ergänzungen.

Schnell komplexe Layouts bauen

Theme-Autor*innen können ihren Themes jetzt schöne Block-Patterns mitgeben, mit denen auch Laien anspruchsvolle Layouts bauen können.

Für die Anwender sind Block-Patterns super-einfach zu handhaben: Vorlage auswählen, Content anpassen, fertig. Wenn man so ein Layout-Modul jedes Mal von Hand erstellen müsste, wäre das ziemlich mühsam.

Block-Patterns sind sehr leicht zu erstellen. Man muss sich nicht mit React, JavaScript und der Gutenberg-API herumschlagen. Jeder, der mit einem Code-Editor umgehen kann, kann diese Funktion nutzen.

Stabiler Demo-Content

Für Theme-Autor*innen haben die Block-Patterns den Vorteil, dass sie damit ihrem Theme standardkonformen Demo-Content mitgeben können, der gut funktioniert. Vorbei die Zeiten, in denen Theme-Autor*innen mit immer neuen Kombinationen von Customizer-Optionen, Shortcodes und Theme-Options herumexperimentieren mussten. Irgendwas funktionierte immer nicht und für die Anwender*innen war es entsprechend schwer, die versprochene Design-Vorlage nachzubilden.

Wir brauchen weniger Custom Blocks

Es gibt unzählige Plugins, über die man sich zusätzliche Gutenberg-Blöcke in die Website holen kann. Meistens handelt es sich um Sets mit mehreren Blöcken. Braucht man nur einen davon, muss man damit leben, dass noch zehn andere Blöcke den Inserter bevölkern, die man überhaupt nicht braucht. Das ist auf die Dauer ziemlich nervig.

Problematisch ist auch, dass viele dieser Plugins das Rad neu erfinden. Ich glaube nicht, dass die Welt 20 Überschriften-Blöcke braucht, die sich jeweils als eigener Block präsentieren, aber in Wirklichkeit nur eine einfache Variante des Überschriften-Blocks sind.

Auch komplexere Layout-Module, wie z. B. Hero-Banner, Testimonials oder Preistabellen, lassen sich sehr gut mit Gutenberg-Bordmitteln und Block-Patterns umsetzen.

Ein speziell dafür programmierter Block ist nicht notwendig. Im Gegenteil. Ein Modul, das in einen Custom Block gegossen ist, ist recht unflexibel. Mal eben eine Kleinigkeit ändern, geht leider nicht. Wird das selbe Module mit einer Block-Pattern umgesetzt, kann man das Modul jederzeit anpassen.

Fazit

Mit Gutenberg werden Layout-Funktionen ins WordPress-System integriert, die man bisher nur mit einem Page-Builder umsetzen konnte.

Ein wichtiger Aspekt dabei ist, dass Gutenberg fester Bestandteil von WordPress ist. Während Page-Builder-Plugins und Theme-Plugin-Pakete, wie z. B. [Avada](#) oder [Divi](#), von freien Entwicklern entwickelt und gepflegt werden, wird Gutenberg vom gesamten WordPress-Ökosystem getragen. Das ist wichtig für die Planungssicherheit. Bei kommerziellen Themes oder Plugins kann es immer sein, dass die Autor*innen ihr Produkt eines Tages nicht weiterführen. Bei Gutenberg besteht diese Gefahr nicht.

Locked-in-Effekt

Ein weiterer Aspekt, der für die Projektplanung wichtig ist: Aus einem Page-Builder kann man nicht so einfach aussteigen. Passt der Page-Builder eines Tages nicht mehr, muss man die Website neu aufbauen und z. B. die Inhalte Seite für Seite per Hand übertragen. Das gilt für alle Fälle, ob man von Page-Builder A auf Page-Builder B umsteigen will oder ob man zu Gutenberg wechseln möchte.

Auch im Gutenberg-Umfeld ist ein Theme-Wechsel mit Arbeit verbunden. Mal eben schnell ein neues Theme aktivieren und alles ist gut, das klappt eher selten.

Aber alle Gutenberg-Blöcke funktionieren mit jedem standardkonformen Theme. Nach dem Theme-Wechsel ist noch alles da. Keine Inhalte gehen verloren, man muss lediglich das Design anpassen.

Reise in die Zukunft mit leichtem Gepäck

Gutenberg ist schlank und modern. Die HTML-Struktur ist weit weniger komplex und verschachtelt als die eines Page-Builders. Für schnelle Ladezeiten ist das ein unschätzbarer Vorteil.

Mit einer Kombination von Block-Styles und Block-Patterns kommt man sehr weit. Man kann damit schöne, individuell gestaltete Websites umsetzen.



Der Autor

Matthias Kittsteiner

Matthias Kittsteiner ist gelernter Mediengestalter aus Stuttgart und arbeitet als Webentwickler seit 2016 intensiv mit WordPress. Dabei erweitert er die Funktionalität für seine Kunden, verbessert deren Website-Ladezeit oder erstellt schlanke und performante Themes. Als Polyglot verbringt er zudem einen Teil seiner Freizeit damit, WordPress selbst und Plugins/Themes auf WordPress.org zu übersetzen.

Full-Site Editing – ein Ausblick ins künftige WordPress

Full-Site Editing ist Teil der Gutenberg-Phase 2, zusammen mit Block-Vorlagen, dem Block-Verzeichnis und blockbasierten Themes. Um den Benutzer*innen mehr Möglichkeiten zu geben, ihre Websites zu verändern, wird dabei ein System geschaffen, um Templates direkt im Block-Editor anzupassen. Auch werden Bereiche wie die Menü-Verwaltung in Blöcke verschoben (mit Hilfe des Navigations-Blocks), so dass mehr und mehr Inhalte aus dem Customizer wegfallen.

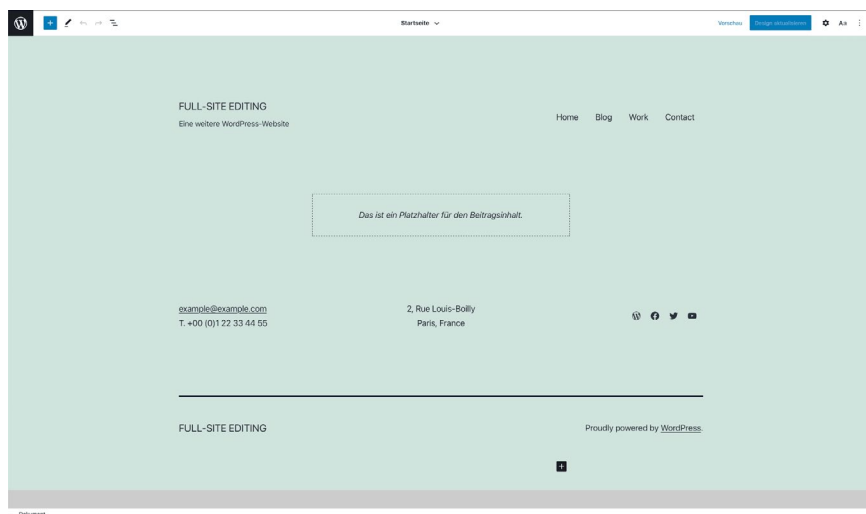
Nachfolgend möchte ich einen aktuellen Überblick präsentieren, wie Full-Site Editing im Moment aussieht, was es bringt und was das für uns alle bedeutet.

Disclaimer

Full-Site Editing (oder auch einfach Site Editing) ist nach wie vor in aktiver Entwicklung. Dies ist eine Momentaufnahme und spiegelt nicht das fertige Full-Site Editing wider. Bis zur Veröffentlichung innerhalb des WordPress Core können sich daher immer noch Dinge ändern. Bisher ist geplant, dass Full-Site Editing mit WordPress 5.7 Einzug in den Core hält (es war aber auch mal für WordPress 5.6 geplant und kann sich daher auch noch weiter verschieben).

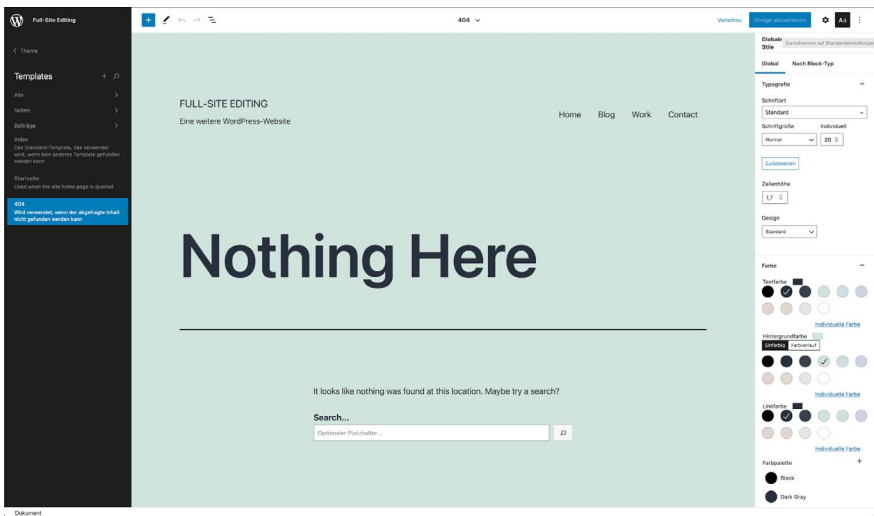
Was ist Full-Site Editing?

Full-Site Editing ist eine Weiterentwicklung des Gutenberg-Editors (Block-Editor), wodurch nicht mehr nur einzelne Inhalte in Form von Beiträgen oder Seiten bearbeitet werden, sondern alle Templates und Template-Bereiche von WordPress. Also ein Großteil dessen, was bisher von einem Theme vordefiniert wurde.



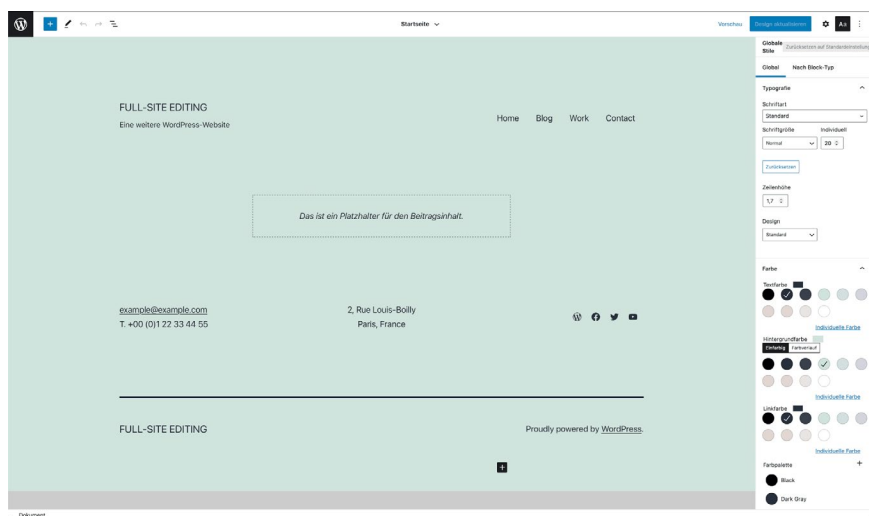
Und was bringt mir das?

Gerade für Nicht-Theme-Entwickler*innen bringt Full-Site Editing eine Menge Vorteile. So lassen sich noch mehr Bereiche der Website direkt im WordPress-Admin-Bereich anpassen, die vorher strikt vom Theme vorgegeben wurden. Ein gutes Beispiel ist hier die 404-Seite. Sie lässt sich, wie alle anderen Seitentypen auch, im Block-Editor genauso anpassen wie jede selbst angelegte Seite auch. Aber auch Bereiche wie Header und Footer können direkt beeinflusst werden. Und durch neue Blöcke wie dem Navigations-Block können Menüs auch im Block-Editor bearbeitet werden.



Was bedeutet das für uns alle?

Themes werden in Zukunft zu großen Teilen verschwinden bzw. für die Administration nahtlos mit anderen Teilen verschmelzen. So kennt man bereits jetzt mit dem Block-Editor die Ansicht zum Bearbeiten von Inhalten und dessen Seitenleiste. Diese wird bei Full-Site Editing beispielsweise auch für globale Einstellungen verwendet, sodass viele Bereiche des Customizers dorthin verschoben werden. Dies bedeutet in einer zukünftigen Version dann den Wegfall des Customizers selbst für diese neue Art von Themes, da er redundante Einstellungen beinhalten würde.



Somit übernehmen Blöcke nicht mehr nur die Darstellung von Inhalten, sondern sind gleichermaßen für das gesamte Design der Website verantwortlich. Gleichzeitig hat man selbst viel mehr Möglichkeiten, das Design seiner Website anzupassen – ganz ohne tiefere Kenntnisse der einzelnen Techniken, die dafür bisher erforderlich waren.

Änderungen in der Bedienung

Durch mehr Funktionalität ändert sich auch automatisch die Bedienung innerhalb des WordPress-Admin-Bereichs. So erhält man die Möglichkeit, Bereiche wie Header und Footer zu bearbeiten, die vorher fest vom Theme vorgegeben wurden. Dafür erhält man über dem Inhaltsbereich eine Breadcrub-Navigation, die anzeigt, welchen Bereich man gerade bearbeitet. Da das Theme blockbasiert sein muss, besteht so auch beispielsweise der Header aus Blöcken und kann im Block-Editor genauso angepasst werden wie jeder andere Bereich. Dabei sieht es praktisch genauso aus, wie es aktuell bei wiederverwendbaren Blöcken (denn rein technisch betrachtet sind Header und Footer nichts anderes) der Fall ist.

Was bedeutet das für Theme-Entwickler*innen?

Für Theme-Entwickler*innen gibt es auf jeden Fall eine Umstellung. Zuerst einmal erfordert es, dass das Theme selbst blockbasiert ist. Das heißt, es muss vollständig aus Blöcken aufgebaut sein. So muss beispielsweise das gesamte Layout des Headers aus normalen Blöcken bestehen, oftmals vermutlich aus einem Gruppen-Block, darin ein Spalten-Block mit zwei Spalten, auf der einen Seite dann mit dem Logo und auf der anderen mit dem neuen (sich noch in Entwicklung befindlichen) Navigations-Block. Man kann es sich so vorstellen, dass das gesamte Layout der Seite inklusive Header, Footer, Seitenleiste etc. im Block-Editor angezeigt werden muss. Und da dieser nur Blöcke anzeigen kann, müssen auch alle Bereiche mit Blöcken definiert werden.

Außerdem muss man sich bei der Entwicklung des Themes bereits darauf einstellen, dass die Benutzer*innen selbst wesentlich mehr Möglichkeiten bekommen werden. So kann beispielsweise ein Block namens „Post Featured Image“ (Beitragsbild) an beliebiger Stelle eines Beitrags eingefügt werden, anstatt dass es strikt durch das Theme vorgegeben wird und nicht verändert werden kann.

Ein weiterer Aspekt, den es zu beachten gilt, ist die zwingende Auseinandersetzung mit dem Block-Editor. Gerade die Erweiterung mit spezifischen Funktionen, die bereits jetzt im Customizer beispielsweise individuelle Anpassungen benötigt, wird auch im Block-Editor erforderlich sein. Da dieser in React geschrieben ist und nicht für alles eine einfache Bearbeitung ermöglicht, wird man sich für Zusatzfunktionen dort einarbeiten müssen.

Aktuell ist Full-Site Editing an ein kompatibles Theme gebunden und daher praktisch ein „Opt-in“. Das heißt, will man selbst diese Funktionalität nicht anbieten oder seinen Kund*innen nicht zumuten (mehr Änderungen seitens von Kund*innen bedeutet immer auch ein potenziell höherer Support-Aufwand oder auch kaputte Websites), kann man darauf einfach verzichten, indem man seine Themes weiterhin so aufbaut wie bisher. Das bedeutet dann jedoch auch, dass zukünftige Verbesserungen und Innovationen möglicherweise nicht automatisch ins eigene Theme mit einfließen und eventuell sogar selbst implementiert werden müssen.

Für Interessierte, die selbst einmal ein blockbasiertes Theme erstellen möchten, gibt es [im Block-Editor-Handbuch ein Tutorial](#) dafür.

Änderungen im Theme-Aufbau

Um dennoch kurz darauf einzugehen, wie sich ein Theme im Aufbau ändert, zeige ich dies anhand des oben verlinkten Tutorials. Wer sich das Ganze einmal live ansehen will, für den gibt es [verschiedene Themes als blockbasiertes Experiment](#) auf GitHub.

Template-Verzeichnisse

Gab es bisher in Themes meistens einen Ordner `template-parts` mit einzelnen Bereichen für beispielsweise den Inhaltsbereich von Seiten oder Blogbeiträgen, so wird es in blockbasierten Themes die Ordner `block-templates` und `block-template-parts` geben:

```
1  theme
2    |__ block-templates
3        |__ index.html
4    |__ block-template-parts
5        |__ footer.html
6        |__ header.html
7    |__ experimental-theme.json
8    |__ functions.php
9    |__ index.php
10   |__ style.css
```

[Zum Code →](#)

Direkt fällt hier auf, dass die Dateien innerhalb dieser Ordner keine PHP-Dateien mehr sind, also keinerlei Logik beinhalten, sondern statische HTML-Dateien.

Um einen Template-Bereich (aus dem Ordner `block-template-parts`) dann zu referenzieren, also beispielsweise in der `index.html` (früher `index.php`) auszugeben, kann man folgenden HTML-Code verwenden:

```
1  <!-- wp:template-part {"slug":"header","theme":"my-theme"} /-->
```

[Zum Code →](#)

Wie auch im Block-Editor werden zusätzliche Informationen innerhalb von HTML-Kommentaren geschrieben und dann später von WordPress verarbeitet. In diesem Fall soll der Template-Bereich `header` (=slug) aus dem Theme `my-theme` (=theme) angezeigt werden.

Um den Beitragstitel auszugeben, genügt dabei schon

```
<!-- wp:post-title /-->, für den Beitragsinhalt wiederum  
<!-- wp:post-content /-->.
```

Templates

Da auch die Templates selbst mit Blöcken aufgebaut sind, ist es am einfachsten, das Layout vorher selbst ebenfalls im Block-Editor anzulegen und von dort über die Code-Ansicht zu exportieren. Andernfalls müsste man den HTML-Code komplett von Hand schreiben und darauf achten, keine Fehler zu machen, die am Ende sonst innerhalb des Block-Editors zu fehlerhaften Blöcken führen.

(experimental-)theme.json

Eine weitere Änderung betrifft die Konfiguration des Themes. Diese fand bisher meistens innerhalb der `functions.php` des Themes statt. In Zukunft wird das zum Teil in einer `theme.json` passieren, aktuell heißt sie noch `experimental-theme.json`. Über JSON können hier Einstellungen für verschiedenste Bereiche vorgenommen werden, beispielsweise Farbpaletten, Zeilenhöhen etc. Dort werden demnach verschiedene Einstellungen kontrolliert, die wiederum das Aussehen oder die Funktionalität einzelner Blöcke definieren.

Dies kann entweder global geschehen oder aber für jeden Block separat. Letzteres ist bereits jetzt von normalen Blöcken bekannt. Hier können beispielsweise bei Absatz-Blöcken Farbe und Schriftgröße verändert

werden, sofern vom Theme erlaubt. Mit Full-Site Editing kann man solche Änderungen auch global für alle Blöcke oder alle Blöcke eines Typs (beispielsweise alle Überschriften zweiter Ebene (h2)) vornehmen.

Beispiel:

```
1  {
2    "settings": {
3      "defaults": {
4        "color": {
5          "palette": [
6            {
7              "slug": "pale-turquoise",
8              "color": "#d1f1f1"
9            }
10           ]
11         }
12       }
13     },
14     "styles": {
15       "core/heading/h2": {
16         "color": {
17           "text": "#000"
18         }
19       }
20     }
21   }
```

Zum Code →

Die globalen Stile sind dann per CSS-Variablen innerhalb des Themes verwendbar. Aus dem oben angegebenen JSON-Code steht beispielsweise die Variable `--wp-preset--color--background` zur Verfügung und kann mit `var(--wp-preset--color--background)` im CSS eingefügt werden.

Die Stile für einzelne Blöcke werden bei den Blöcken direkt hinterlegt, können dort jedoch noch angepasst werden, sofern die Farbpalette zur Verfügung steht.

Eine [Übersicht über die verschiedenen Eigenschaften](#), die man hier angeben kann, ist im Block-Editor-Handbuch zu finden.

Was passiert mit Page-Buildern?

Bereits mit dem Einzug des Block-Editors in den WordPress Core wurde es für Page-Builder schwieriger. Vieles, das früher nur mit selbigen möglich war, kann man nun in Gutenberg ebenso umsetzen – und zumeist mit einem wesentlich saubereren Quellcode und einer besseren Ladezeit im Frontend.

Verwendet man aktuell bereits einen Page-Builder, ist eine Umstellung auf den Block-Editor im Allgemeinen und Full-Site Editing im Speziellen keine leichte Aufgabe. Im Normalfall heißt das: Alle Inhalte müssen manuell umgezogen werden. Einen Importer oder ähnliches gibt es in der Regel nicht. Dieser „Lock-in“-Effekt betrifft praktisch alle Page-Builder.

Nichtsdestotrotz verändert sich durch Full-Site Editing gar nicht so viel. Denn bereits während der Entwicklung des Block-Editors wurde klar, dass er am Ende funktionell mit den großen Page-Buildern gleichziehen oder

sie sogar überholen wird. Gleichzeitig ist das gesamte Ökosystem hinter dem Block-Editor wesentlich größer, sowohl aus Entwickler-Seite, die explizit am Editor selbst arbeiten und ihn verbessern, aber auch von Seiten unabhängiger Entwickler*innen, die eigene Blöcke und Funktionen in den Block-Editor integrieren.

Letzten Endes unterscheiden sich Page-Builder (inklusive des Block-Editors) vor allem durch ihren Aufbau, identische Grundfunktionalitäten bringen sie alle mit. Daher liegt die Entscheidung bei denjenigen, die das WordPress(-Theme) einrichten, sich zu entscheiden, welcher Editor verwendet werden soll. So wird sich jemand mit wenig Wissen in diesem Bereich oftmals nur aufgrund von Abhängigkeiten (beispielsweise des Themes) für einen anderen Editor entscheiden und ansonsten den Standard verwenden, der vorinstalliert ist – den Block-Editor.

Gerade Agenturen und Auftragsentwickler*innen haben hier die Möglichkeit, den Editor auf Basis der eigenen Wünsche vorzudefinieren. Hier ist es vor allem ein ganz anderer Workflow während der Entwicklung und wie viel Erfahrung man mit dem einen oder dem anderen Editor hat. Dennoch sollte man hier nie die Objektivität außer Acht lassen: Ist der Editor wirklich für meine Kund*innen geeignet? Verlangsamt der Editor nicht den Aufbau der Website?

Insbesondere aus Gewohnheit werden Page-Builder vermutlich noch viele Jahre (über-)leben. Die Entwicklung des Block-Editors schreitet jedoch in einem wesentlich schnelleren Tempo voran, sodass es für Page-Builder irgendwann schwierig sein wird, hier mithalten. Und auch wenn der Mensch ein Gewohnheitstier ist und gern bei dem bleibt, das er kennt, so werden sich nach und nach immer mehr für den Block-Editor entscheiden, gerade auch aufgrund solcher Funktionen wie dem Full-Site Editing.

Und wozu das alles?

Je mehr Bereiche des aktuellen WordPress-Adminbereichs im Block-Editor landen, desto einheitlicher wird das gesamte System rund um die Erstellung von Inhalten und Anpassung des Aussehens der Website. Das hilft am Ende jedem, der/die diese Bereiche bearbeitet, also praktisch jedem, der/die WordPress verwendet.

Außerdem erhalten Inhaltsersteller*innen mehr Kontrolle darüber, wie die Seite oder die gesamte Website aussieht, ohne selbst ein Theme entwickeln oder anpassen zu müssen. Es ist ein demokratischer Ansatz, jedem der/die den WordPress-Adminbereich verwendet – basierend auf seiner/ihrer Rolle – die Verantwortung für das Design zu geben. Und während genau das etwas ist, das Designer*innen und Entwickler*innen oftmals nicht wollen – die Kontrolle des Aussehens abgeben – ist es genau das, was viele Benutzer*innen letztendlich wollen.

Um das in gewissem Sinne zu limitieren, können Designer*innen und Entwickler*innen beispielsweise Farbpaletten vorgeben, die dann standardmäßig angezeigt werden und Benutzer*innen mit einem Klick die Möglichkeit geben, eine vordefinierte und dadurch passende Farbe zu wählen.

Letzten Endes ist Full-Site Editing ein weiterer Schritt auf dem Weg, dass mehr Personen mehr Möglichkeiten haben, ohne sich mit den technischen Hintergründen dahinter auseinanderzusetzen und die Website so darzustellen, wie sie es möchten.



Der Autor

Ralf Wiechers

Die Vielseitigkeit und Abwechslung in den Projekten, die mit WordPress und WooCommerce umsetzbar sind, begeistern den studierten Kommunikationsdesigner [Ralf Wiechers](#) immer wieder. Als ortsunabhängiger digitaler Nomade berät und unterstützt er Firmen bei der Umsetzung von individuellen Websites, die nicht nur gut aussehen, sondern auch was können.

Wie WooCommerce plus Gutenberg das Online-Einkaufserlebnis verändern wird

Der Trend zum E-Commerce ist seit vielen Jahren ungebrochen. Die weltweite Corona-Pandemie hat diesen nur noch sprunghaft beschleunigt. Ebenso sind viele weitere Veränderungen, die mal klein angefangen haben, inzwischen groß geworden, ohne dass man sie wahrgenommen hat.

Wenn man nach dem Gefühl fragt, wie viele Benutzer*innen mobil auf Website oder Shop zugreifen, sagen Betreiber und Agenturen gerne mal so etwas wie „vielleicht 10 Prozent“. Doch hat sich der Anteil von „mobilen Geräten“ beim Internet-Traffic im Jahr 2020 auf stolze 45 Prozent hochgeschraubt. Es wurde schon für viel, viel kleinere Marktanteile optimiert, zum Beispiel für den Internet Explorer bei unter 5 Prozent.

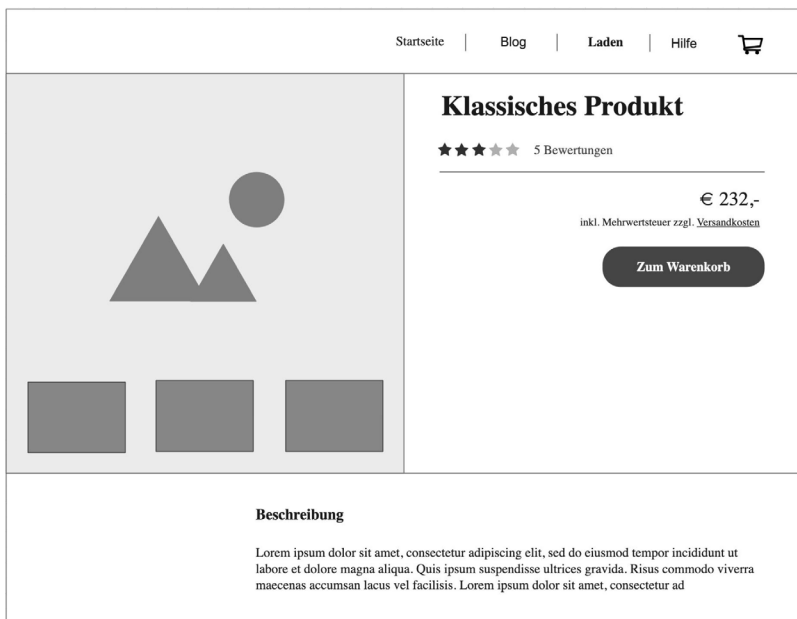
Auch sind Anbieter wie [Wish](#), die ausschließlich mobil mit eigenen Apps arbeiten, inzwischen sehr erfolgreich.

Um für die sich verändernden Erwartungen und technischen Möglichkeiten in der Zukunft gewappnet zu sein, wurde in WordPress der Block-Editor eingeführt. Dieser Artikel betrachtet den heutigen Stand von Produktseiten in Onlineshops und welche Möglichkeiten und Verbesserungen mit Gutenberg möglich werden, um den Einkauf online noch informativer und interessanter zu gestalten.

Altbackende Shops

Während klassische Onlineshops gerade scheinbar Ladengeschäfte revolutionieren, werden sie selber momentan von neuen E-Commerce-Angeboten überholt.

Das klassische Layout einer Produkt-Detailseite enthält ein oder mehrere Produktbilder, daneben Name, Artikelnummer, Preis und Warenkorb-Button. Darunter die keyword-reiche Beschreibung. In Tabs oder im Akkordeon versteckt die technischen Daten und das Kleingedruckte. Jetzt noch die lange, langweilige Liste der Bewertungen. Abschließend die irgendwie verwandten Produkte automatisch verlinkt.



Dieses Standardlayout wurde für den großen Bildschirm am Schreibtisch entwickelt und stumpf mit Farben und Schriftarten über alle Arten von Produkten und Branchen ausgebreitet.

Egal, ob von der Baustelle oder dem Sofa aufgerufen, das Layout macht auf Smartphones keine gute Figur. Besonders der „Above the fold“-Ansatz sprengt jetzt den kleinen Bildschirm (plus Cookie-Banner und Newsletter-Pop-up). Oft werden alle Infos als super wichtig deklariert und ganz nach oben geschrieben. Alles, was die Hürde nicht nimmt, wird irgendwo noch knapp notiert oder einfach komplett weggelassen.

Der neue Stil

Ein Konzept, das sich im Online-Marketing schon lange erfolgreich etabliert hat, ist die „Verkaufsseite“ oder auch „Landingpage“ oder „Leadpage“. Diese Seiten sind oft erstaunlich lang. Aber ihr Inhalt ist sehr genau auf eine Zielgruppe und ein einzelnes Produkt zugeschnitten. Wenn eine Interessentin über eine Anzeige oder das Long-Tail-Keyword direkt auf der Produktseite gelandet ist, ist auch die Bereitschaft zum Scrollen da. Der Finger ist vom minutenlangen Scrollen in Timelines meist gut trainiert.

Die Motivation, den Personen hinter dem Shop/Angebot so viel Arbeit wie möglich zu ersparen, deckt sich mit dem Interesse des potentiellen Kunden. Es werden alle Vorteile des Produktes umfangreich heraus- und die Details klar dargestellt. Mögliche Fragen werden bewusst formuliert und samt Antworten offensiv eingebaut.

Das erklärte Ziel ist es, den Interessenten so gut zu informieren, dass keine Zweifel mehr am Produkt bestehen, damit am unteren Ende der Seite unverzüglich der Kauf vollzogen wird. Denn ein Interessent, der nochmal woanders nach einem Detail schauen oder auf die Antwort vom Support warten muss, ist oft ein verlorener Kunde.

Dabei wird auf alles zurückgegriffen, was das moderne Web so kann, um Textwüsten zu vermeiden.

- Mit Fotos, Visualisierungen und Videos wird das Produkt und der Hersteller/Lieferant ins rechte Licht gerückt.
- Wer kann und hat, bietet 360-Grad-Ansichten oder „Erweiterte-Realitäts“-Elemente an.
- Bewertungen von früheren Verkäufen werden mit Porträt und Name groß dargestellt.
- Verschiedene Vertrauenselemente wie Siegel oder Auszeichnungen werden eingebunden.
- Mit Icons, Animationen, Überschriften, Spalten und anderen Layout-Methoden werden optische Highlights erzeugt.

Auf dieser Art von Seite findet man bei manchen Anbietern auch Dark-Patterns oder psychologische Tricks. Diese sind aber nicht zwangsläufig, sondern bewusst eingesetzt.

Beratermangel ausgleichen

Im Ladengeschäft kann im persönlichen Gespräch auf Fragen direkt eingegangen werden. Diesen Mangel an direktem Kontakt muss mit größerem Informationsangebot begegnet werden.

Fotos sagen vielleicht mehr als viele Worte. Das Produkt aber nur im unendlichem Weiß aus x Winkeln zu sehen, gibt kein Gefühl für Größe, Gewicht, Anmutung und Funktion. Mit einer Dokumentation, die schon fast als Bedienungsanleitung durchgehen könnte, kann sich ein deutlich besseres Bild gemacht werden. So lässt sich bei Möbelhäusern mit DIY-Konzept schon vor dem Kauf die echte Anleitung ansehen.

Gerade Personen, für die der digitale Einkauf neu ist, benötigen dies, um unnötige Bestellungen zum „Nur mal angucken“ zu vermeiden.

Bonus-Vorteil: Wenn technisch gut gemacht, mögen diese Inhalte auch die Suchmaschine gerne.

Neue Felder machen es vor

Auch streben immer mehr neue Produktformen auf den Markt. So ist das Abo stark im Kommen. Egal, ob der Zugang zu Trainingsgeräten zum monatlichen Fixpreis, die Online-Community passend zum Brauerei-Hobby oder ein digitales Werkzeug für die Buchhaltung.

Nicht ganz neu, aber auch vermehrt anzutreffen, sind Online-Buchungs- und Reservierungs-Produkte. Diese personalsparende Verkaufsform bietet unmittelbares Feedback und wird von Kunden zunehmend bevorzugt.

Die neuen Produkte sind auch Teil des E-Commerce und können mittels [WooCommerce](#) verkauft werden.

Die Produktseiten dieser Produkte sind umfangreich und rufen regelmäßig in ihrem Verlauf zum Kaufen auf. Ein Prozess, der unmittelbar mit dem nächsten Klick gestartet wird. Sie sind weit entfernt von den klassischen Shop-Layouts mit computer-generierten Boxen als Produktbild für eine Software oder andere virtuelle Güter. Waren diese nicht immer schon lächerlich?

Schöne Fokussierung wagen

Das Layout einer Produktseite muss neu erdacht werden. Klare und ansehnliche, untereinander angeordnete Informationen lassen sich viel besser konsumieren und sorgen für einen längeren Aufenthalt, da sich in Ruhe mit dem Produkt auseinandergesetzt werden kann. Dieser Vorteil gilt auch für größere Bildschirme.

Wenn beim Lesen der Produktseite festgestellt wird, dass es das falsche Produkt ist, wird oft direkt die „Zurück“-Funktion genutzt, um wieder in der Suchmaschine oder Listenansicht zu landen. Mit gezielt platzierten Links kann dort, wo ein Interessent feststellt, dass es nicht passt, gezielt abgefangen werden. So ist ein „Nicht das passende Modell?“ oder „Funktion X benötigt? Siehe hier“ deutlich hilfreicher, führt schneller ans Ziel und verhindert den Absprung. Das visuelle Herumgezerre, doch unbedingt auf dieser Seite weiter zu lesen, ist einfach nur im Weg.

Starter	Kleinunternehmen	Mittelstand
		
1 Account	5 Accounts	30 Accounts
€59 ^{pro Monat}	€99 ^{pro Monat}	€149 ^{pro Monat}
Loslegen	Loslegen	Loslegen

Mehr Accounts benötigt?

[Unsere Enterprise-Tarife](#)

Eine Zukunft mit Blöcken

Dank der Blöcke, die mit Gutenberg eingeführt wurden, ist die Erstellung von schöneren Inhalten direkt mit dem WordPress-Core deutlich einfacher geworden. Aktuell noch in der Entwicklung, wird der Block-Editor auch bald für Produkte in WooCommerce verfügbar sein.

Der aus meiner Sicht größte Vorteil, den WordPress und WooCommerce haben, ist die Kombination aus CMS und Shop-System in einem Werkzeug. So lassen sich durch das CMS schöne Inhalte bauen, die auch gleichzeitig die Produktseite sind. Das ist die große Schwachstelle von bisherigen Landingpage-Angeboten. Die Verkaufsseite wurde auf einem System nur für hübsche Seiten gebaut. Der eigentliche Verkauf muss aber bei einem weiteren Anbieter vollzogen werden. Datenschutz mal ganz beiseitegelassen.

Selbst große Elektronik-Konzerne schaffen es nicht, die emotionalen Marketing-Seiten direkt im Shop-System umzusetzen. Wenn einen die Produktpräsentation überzeugt hat und man auf „Jetzt Kaufen“ klickt, landet man allzu oft in einem recht unschönen Shop-Bereich auf einer Subdomain.

Dies ist eine Bruchkante, an der Vertrauen und Nutzer*innen verloren gehen.

Kontrolle vs. Emotionen

Der Grund für die unterschiedlichen Systeme bei vielen Anbietern sind die unterschiedlichen Wünsche der Shop-Verantwortlichen und der Marketing-Abteilung. Die technische Seite möchte ein langlebiges System, um Zahlungsanbieter, Logistik und Buchhaltung stabil anbinden zu können.

Schließlich ist Downtime im Shop bares Geld wert. Im Marketing-Bereich will man sich schnell bewegen können, um Kampagnen und Aktionen schnell und regelmäßig anzupassen.

Die Lösung waren bisher meist separate Websites für beide Bereiche oder technisch labile iFrame- sowie Embed-Lösungen. Hier können die Gutenberg-Blöcke einen echten Unterschied machen.

Kontrollierte Flexibilität

Mit dem Block-Editor lässt sich ein „In den Warenkorb“-Bereich in einer Produktseite einbinden und flexibel platzieren. Durch Block-Templates lassen sich so Pflichtblöcke festlegen und gleichzeitig viel Layout-Freiraum gewähren. Diese Pflichtblöcke sind vor dem Entfernen geschützt. Sie werden bei der Anlage eines neuen Produktes automatisch platziert und können so nicht vergessen werden. So kann eine tolle Produktseite gelayoutet werden, ohne dass vorher alle Positionen festgelegt sind. Mit etwas Schulung kann große Freiheit an die Gestaltenden gegeben werden, so lange sie z. B. keine Preise direkt eingeben, sondern „Variablen“ oder spezielle Blöcke verwenden.

Durch diese speziellen Blöcke können auch rechtliche Vorgaben (wie etwa die Steuerinformation bei Preisen in Shops für den deutschen Raum) code-basiert sichergestellt werden.

Gerade für Shops, die an Warenwirtschaftssysteme oder andere Schnittstellen angeschlossen sind, haben derartige Platzhalter, die Post-Meta-Felder oder Produkteigenschaften ausgeben, eine große Bedeutung. Das automatische Ändern von Produktdetails oder Lagerbeständen ist eine wichtige Funktion für effiziente und zuverlässige Prozesse.

Neben Pflichtblöcken kann der Block-Editor noch weitere Aufpasserfunktionen übernehmen. So können Blöcke, die z. B. im Blog-Post erlaubt sind, für Produkte verboten sein. Auch die Nachbarschaft oder Verschachtelung von bestimmten Blöcken kann erzwungen oder unterbunden werden.

Auch wenn es im ersten Moment so aussieht, als ob der Block-Editor dazu führt, dass die Shop-Technik Kontrolle abgibt, ist die Realität anders. Es bestand vorher nur geringe Kontrolle über das, was in der Produktbeschreibung stattfindet. HTML-Snippets oder Shortcodes waren schwer zu beeinflussen und wurden deswegen gerne mal knallhart weggefiltert. Mit der Unterteilung der Beschreibung bzw. der Produktseite in Blöcke lässt sich deutlich einfacher programmatisch zwischen oder auf einzelne Elemente zugreifen und Änderungen mittels Plugins vornehmen.

Typografie im Griff

Gerade Nutzer*innen, die von Page-Buildern kommen, sind es gewohnt, jede Kleinigkeit von jedem einzelnen Element/Block einstellen zu können. Eine Vielzahl der Möglichkeiten bietet Gutenberg inzwischen auch. Diese Freiheit ist in manchen Shops gewünscht und in anderen Shops eine gruselige Vorstellung. Zum Glück bietet der Block-Editor die Möglichkeit, hier sehr granular Einstellungsmöglichkeiten freizugeben bzw. sie zu unterbinden.

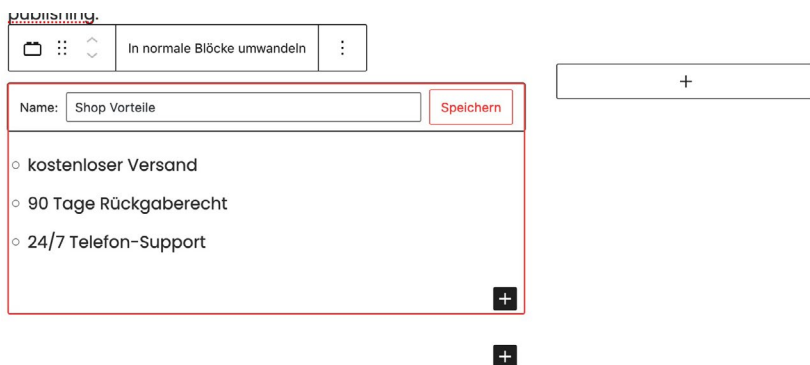
Somit kann nicht nur eine Balance beim Layout, sondern auch bei der Typografie zwischen Corporate-Identity und Gestaltungsfreiraum ganz an die Bedürfnisse des Projekts angepasst werden.

In dem Zusammenhang ist die Zukunft der Themes auch noch ein interessantes offenes Thema.

Wiederholen mit System

Eigentlich ist das Kernprinzip eines CMS „Einmal eingeben | x-fach ausspielen“. Doch in vielen WordPress-Websites ist dies bisher nicht bewusst genutzt worden. In größeren Projekten musste man es durch komplizierte Lösungen mit Plugin-Abhängigkeiten umsetzen. WordPress realisiert nun eine wichtige Funktion auf einem visuellen und gut nutzbaren Weg.

Es lassen sich aus beliebigen Inhalten mit zwei Klicks „wiederverwendbaren Blöcke“ erstellen, die anschließend in andere Seiten oder Produkte 1:1 wiederverwertet werden können. Dabei bleiben die Inhalte aber verbunden. Eine Änderung in einem der platzierten Instanzen aktualisiert auch die anderen. Dabei richtet sich die Spiegelung nur nach innen. Die Platzierung auf der Seite ist frei wählbar.



Dies ist eine Funktion, die Produktseiten deutlich besser wartbar macht. Es gibt Inhalte, z. B. Versanddetails oder Werbeflächen, die auf einigen oder allen Produktseiten identisch sein sollen. Ihre Position auf der Seite aber nicht.

Volle Bandbreite

Als das am schnellsten wachsende Shop-System ist WooCommerce aktuell im Bereich von „kleineren Shops“ weit verbreitet. Diese tendieren zu einer geringen Anzahl an Produkten, die aber mit Blöcken und deren Möglichkeiten umso liebevoller und detailreicher von ihren Produzierenden präsentiert werden können.

Für Shops mit vielen Produkten bedeutet es aber noch lange nicht das Aus oder dass der Block-Editor ihnen komplett das Leben schwer macht. Mit den Fähigkeiten der Entwicklungsabteilung, die bei jedem großen Shop-Projekt so oder so Pflicht ist, lassen sich ebenfalls tolle Funktionen herausholen.

Die Bandbreite, mit wenigen Produkten zu starten und damit richtig groß werden zu können, macht das Content-Management-System mit dem Shopping-Plugin sehr attraktiv.

Tolle Kombination: WooCommerce plus Gutenberg

Ähnlich wie für Entwickler*innen im WordPress-Bereich steht den selbigen mit der Spezialisierung auf WooCommerce einiges an neuer Technik und veränderten Konzepten ins Haus.

Auch ist ein gewisser Kampf um Bedeutung und Bewahrung von Altem absehbar. Wie viel von den alten Templates oder bisherigen Hooks und Filtern weiter Bestand haben werden, ist noch nicht klar.

Die kommenden Gutenberg-Blöcke für WooCommerce sind eine Änderung, auf die sich Shop-Betreibende und Käufer gleichermaßen freuen können. Sie macht die Erstellung von schöneren, individuellen und informativen Produktseiten deutlich leichter – egal wie groß der Shop ist.



Der Autor

Simon Kraft

Simon Kraft ist seit 2008 WordPress-Entwickler. Er ist ein aktives Mitglied der WordPress-Community und im Pluginkollektiv. Als Speaker und Berater beschäftigt er sich mit Performance-Optimierung und den Auswirkungen des Internets auf den Klimawandel.

Gute WordPress-Plugins erkennen und auswählen

Plugins erlauben es Ihnen, WordPress mit wenigen Klicks um die verschiedensten Funktionen zu erweitern. Nun gibt es neben dem offiziellen Plugin-Verzeichnis auf WordPress.org mit aktuell über 58.000 Plugins ein unüberschaubares Angebot an kostenpflichtigen und kostenlosen Plugins. Einsteiger*innen geraten hier bei der Plugin-Auswahl schnell in ernste Probleme. Aber auch Profis können schon angesichts der bloßen Menge an verfügbaren Plugins ins Straucheln kommen.

Im Folgenden möchte ich Ihnen ein paar grundlegende Tipps an die Hand geben und Sie damit in die Lage versetzen, bei Plugins zukünftig selbst die Spreu vom Weizen zu trennen und grobe Fehltritte zu vermeiden. Dafür sind wir im Übrigen nicht darauf angewiesen, einen Blick auf den Programm-Code der Plugins zu werfen. Besonders bei Bezahl-Plugins ist das vor dem Kauf in vielen Fällen ohnehin nicht möglich.

Mögliche Quellen für Plugins

Die unzähligen Quellen, aus denen Sie Ihre Plugins beziehen können, lassen sich in einige grundlegende Kategorien einteilen. Diese sollten wir kurz mit ihren jeweiligen Vor- und Nachteilen anreißern.

WordPress.org



Das Plugin-Verzeichnis auf WordPress.org enthält über 50.000 kostenfreie Plugins.

Alle Plugins, die sich im offiziellen Plugin-Verzeichnis finden, haben eine große Gemeinsamkeit: sie sind kostenfrei. Das Verzeichnis erreichen Sie entweder direkt über de.wordpress.org/plugins oder aus Ihrem WordPress-Backend heraus über das Admin-Menü und den Punkt **Plugins / Installieren**.

Bei der Aufnahme auf WordPress.org werden Plugins eingehend von einem Team Freiwilliger geprüft und bei Beanstandungen zur Verbesserung an den/die Plugin-Autoren*in zurückgeschickt. Plugins mit bekannten Sicherheitsproblemen werden umgehend aus dem Angebot genommen.

Nach der Installation können Plugin-Updates später einfach mit einem Klick aus Ihrem WordPress-Backend heraus oder sogar komplett automatisch durchgeführt werden.

Beachten sollten Sie hier allerdings, dass viele der Entwickler*innen in der Regel keinen professionellen Support auf WordPress.org bereitstellen. Die Support-Foren sind also eher als Selbsthilfegruppen zu verstehen.

Entwickler-Seiten

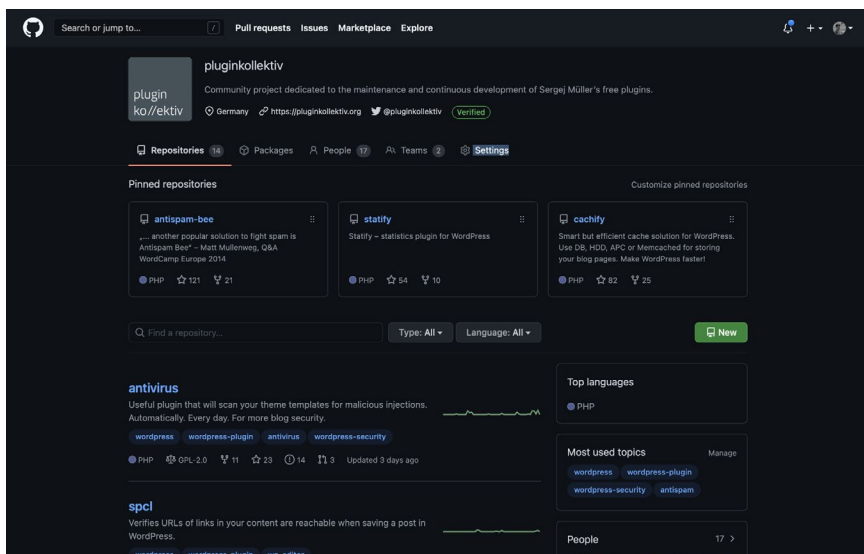
Viele Entwickler*innen, ganz unabhängig vom Umfang ihrer Tätigkeit, bieten Ihre Plugins auf einer eigenen Website an. In den Fällen größerer Anbieter können hier sogar eigene Shops mit verschiedenen Produkten und Services entstehen. Berühmte Beispiele für solche Entwickler-Shops sind woocommerce.com, deliciousbrains.com oder marketpress.de.

Bei diesem Modell haben die Anbieter deutlich mehr Freiheiten und können unabhängiger von den Regeln des WordPress.org-Verzeichnisses agieren. Mit allen Vor- und Nachteilen, die das für Sie als Kunde mitbringt.

Marktplätze

Anbieter, die den Betrieb eigener Infrastruktur scheuen, finden sich in der Regel auf Marktplätzen wie [CodeCanyon](https://codecanyon.com), die die Plattform gegen eine Umsatzbeteiligung für den Verkauf und Support von Plugins und anderen Produkten bereitstellen.

GitHub



GitHub ist sowohl Entwicklungsplattform für Plugin-Entwickler*innen als auch Quelle für Plugins.

Die Code-Plattform [GitHub](#) dient Entwickler*innen vorrangig als gemeinsame Arbeitsumgebung und Ticket-System, wird aber auch gern zum kostenlosen Bereitstellen kleiner Plugins und Fingerübungen genutzt. Zusätzlich finden sich hier viele Plugins, die auch auf WordPress.org verfügbar sind. Ein Blick auf die GitHub-Seite eines Plugins lohnt sich außerdem fast immer, weil Sie hier schnell ein Gefühl dafür bekommen können, ob ein Plugin aktiv weiterentwickelt wird, oder nicht.

Grundsätzlich gibt es keine zwingenden Argumente für oder gegen eine dieser Quellen. Genauso halte ich es für wenig zielführend, kostenfreie oder Bezahl-Plugins grundsätzlich zu bevorzugen. Sie sollten nur in jedem Fall darauf achten, Plugins nur von den tatsächlichen Autoren zu beziehen. Besonders bei den kostenpflichtigen Lösungen kommt es gerne einmal vor, dass Drittanbieter günstige Bundle-Angebote schnüren oder Plugins sogar kostenlos weitergeben. Nicht selten enthalten diese Angebote Schadcode, der Ihre Website in Gefahr bringen kann.

Checkliste

Keiner der folgenden Punkte ist für sich allein genommen das perfekte Entscheidungskriterium. Dennoch können Sie mit jedem der Punkte Ihr Gesamtbild eines Plugins etwas weiter vervollständigen. Wenn für Ihren Anwendungsfall mehrere Plugins infrage kommen, empfiehlt es sich tatsächlich, eine kleine Liste anzulegen und für jeden der folgenden Aspekte Plus- bzw. Minuspunkte zu vergeben.

Funktionsumfang

Ein Plugin nach dem Funktionsumfang zu bewerten, mag im ersten Moment nach einer Binsenweisheit klingen. Tatsächlich meine ich damit gleich zwei leicht unterschiedliche Dinge.

Natürlich ist es wichtig, dass ein Plugin, das Sie einsetzen werden, Ihre Anforderungen abdeckt. Wollen Sie zum Beispiel T-Shirts verkaufen, würde Sie ein Online-Shop-Plugin ohne Versand-Logik wenig glücklich machen. Daher ist es nützlich, wenn Sie sich vorab über Ihre Anforderungen im Klaren sind.

Gleichzeitig halte ich es aber für mindestens genauso wichtig, dass Sie ein Plugin auswählen, das nach Möglichkeit auch nicht viel mehr abdeckt als den benötigten Funktionsumfang. So sollte das Plugin für Ihren Online-Shop nicht auch noch eine Newsletter-Funktion mitbringen. Wenn Sie keinen Newsletter benötigen, sollte der Code dafür auch nicht auf Ihrer Seite sein. Und sollten Sie tatsächlich einen Newsletter einsetzen wollen, wären Sie mit Sicherheit mit einem dedizierten Newsletter-Plugin besser bedient.

Letzte Änderungen

Version: **2.9.3**

Zuletzt aktualisiert: **vor 1 Monat**

Aktive Installationen: **600.000+**

WordPress-Version: **4.5 oder höher**

Getestet bis: **5.6**

PHP-Version: **5.2 oder höher**

Sprachen: **[Zeige alle 26](#)**

Schlagwörter: [anti-spam](#) [antispam](#)

[block spam](#) [comment](#)

[comments](#)

[Erweiterte Ansicht](#)

WordPress.org gibt detaillierte Informationen über letzte Updates, die aktuelle Version und andere Eckdaten eines Plugins.

Wie lange die letzte Aktualisierung eines Plugins zurückliegt, sagt im ersten Moment nicht viel über dessen Funktionstüchtigkeit aus. Es ist durchaus denkbar, dass ein Plugin unter Optimalbedingungen Jahre lang ohne Update funktioniert, auch wenn WordPress selbst regelmäßige Neuerungen erfährt und sich aus Anwendersicht extrem verändert.

Dennoch ist ein Blick auf die letzte Aktualisierung eines Plugins für uns lohnenswert. Zum einen habe ich eben aus gutem Grund von **Optimalbedingungen** gesprochen – in der Praxis ist das nämlich häufig nicht der Fall. Zum anderen können wir vom letzten Änderungsdatum darauf schließen, ob das betreffende Plugin noch aktiv betreut wird. Wenn hier zumindest jeweils die Kompatibilität mit der aktuellen WordPress-Version verkündet wird, können wir davon ausgehen, dass das Projekt nicht ganz tot ist.

Im besten Fall finden wir ein Changelog, in dem sich gleich mehrere zurückliegende Aktualisierungen einsehen lassen. Das kann uns unter Umständen noch ein etwas besseres Gefühl für den Umfang und die Regelmäßigkeit von Änderungen geben.

Support-Aktivitäten

Gibt es ein öffentliches Support-Forum, [wie auf WordPress.org](https://wordpress.org/support/), so lohnt sich auch hier ein Blick. Viele unbeantwortete Anfragen können ein Indikator dafür sein, dass ein Plugin nicht aktiv betreut wird, dass der/die Entwickler*in sich nicht mehr dafür interessiert oder dass die Entwicklung komplett eingestellt wurde.

Dem muss aber nicht so sein. Zumindest auf WordPress.org erfolgt der Support auf freiwilliger Basis, einige Entwickler*innen haben kein Interesse

daran, das Forum für Support zu nutzen. Das ist nicht unbedingt ein Ausschlusskriterium, verrät Ihnen aber, womit Sie rechnen müssen, wenn Sie selbst einmal eine Frage haben.

Auf kommerziellen Plattformen wird Ihnen vor dem Kauf nicht immer die Möglichkeit zur Verfügung stehen, ein Forum einzusehen. Vor dem Kauf lohnt es sich dann, mit dem Support Kontakt aufzunehmen und eine Frage zu stellen. Reaktionszeit, Kooperationsbereitschaft, Freundlichkeit und vielleicht sogar Sprache der Support-Mitarbeiter könnten für Sie in der Zukunft relevant sein.

Beschreibung und Auftritt

Wie fühlen Sie sich, wenn Sie den Auftritt eines Plugins betrachten? Werden Ihnen Funktionen und Umfang aus der Beschreibung klar? Sind relevanten Fragen beantwortet? Lässt das Design der Seite (sofern es sich nicht um WordPress.org oder eine der großen Plattformen handelt) auf einen professionellen Anbieter schließen oder doch eher auf ein Hobby-Projekt?

Nach solchen Äußerlichkeiten zu entscheiden, ist natürlich nicht sonderlich objektiv. Aber mehr als einmal habe ich mich schon gegen ein Plugin entschieden, dessen Website absolut unprofessionell wirkte und mich wirklich daran zweifeln ließ, ob es um den Zustand des Plugin-Codes besser bestellt ist.

Autor

Profiles

[Logged in as: Simon Kraft](#)
[Log Out](#)



Simon Kraft
 @kraft on WordPress.org and Slack

Contribution History

 Meetup Organizer
  Plugin Developer

 Translation Contributor
  Translation Editor

 WordCamp Organizer
  WordCamp Speaker

Member Since: August 6th, 2011

Location: Germany

Website: simonkraft.de

Employer: YITH

Find me on: 

Activity
 Plugins
 Favorites
 Translations

Developer


Impressum
 ★★★★★
 Active installs: 3,000+

Contributor


Antispam Bee
 ★★★★★
 Active installs: 600,000+


Block Control
 ★★★★★
 Active installs: 20+


Embed Privacy
 ★★★★★
 Active installs: 200+

Für Entwickler*innen auf WordPress.org vermitteln die dortigen Benutzer-Profile einen guten ersten Eindruck.

Auch wenn Sie die Namen bekannter WordPress-Entwickler*innen nicht im Schlaf aufsagen können, kann es hilfreich sein, einen Blick auf den oder die Urheber eines Plugins zu werfen.

Im ganz banalen Fall starten Sie eine Google Suche mit dem Namen und dem Zusatz „WordPress“. Auch hier ist ein Stück weit Ihr Bauchgefühl gefragt. Wirkt die betreffende Person so, als wisse sie, wie man WordPress buchstabiert? Bonuspunkte gibt es für ein ordentlich gepflegtes Blog mit Fachartikeln, Vorträgen auf WordPress-Meetups oder anderen Veranstaltungen. Viele Entwickler*innen haben zudem

ein WordPress.org-Profil, in dem sich ihre Aktivitäten und Errungenschaften zumindest erahnen lassen. Hier ist besonders der „Plugins“-Abschnitt interessant. Hat der betreffende Autor weitere Plugins im Sortiment? Sind sie gut bewertet und häufig im Einsatz?

Ist der Urheber Ihres Plugins hingegen gänzlich unbekannt, oder tritt als schwer nachverfolgbares Pseudonym in Erscheinung, ist ein wenig mehr Vorsicht geboten.

Bewertungen

Die Art (und Qualität) von Bewertungen, die Ihnen zur Verfügung steht, unterscheidet sich je nach Plattform deutlich. Dennoch geben auch diese uns einen weiteren wertvollen Anhaltspunkt in unserer Bewertung.

Auf WordPress.org finden Sie eine einfache Bewertung von 1 bis 5 Sterne. Neben der offensichtlichen Sterne-Anzahl sollten Sie hier vor allem auch auf die Anzahl der Bewertungen achten. Eine perfekte 5-Sterne-Wertung ist einfach erreicht, wenn nur der/die Entwickler*in selbst sein/ihr Plugin bewertet hat. Wenn dutzende oder gar hunderte Bewertungen in die Berechnung einfließen, sind glatte fünf Sterne deutlich beeindruckender. Auch ein genauerer Blick auf einige der Bewertungen lohnt sich. Nicht immer handelt es sich um fundierte Bewertungen der Plugin-Funktionen. Eine schlechte Bewertung muss also nicht immer ein Ausschlusskriterium sein.

Wird das betreffende Plugin auf einem anderen Marktplatz angeboten, finden Sie auch dort in der Regel Bewertungen, die denen auf WordPress.org in den Grundzügen gleichen.

Bieten Entwickler*innen die Plugins auf ihren eigenen Websites an, dürfen wir nicht vergessen, dass wir nur sehen, was der/die Entwickler*in uns sehen lässt. Aber auch hier können wir auf eine einfache Google-Suche zurückgreifen, um Erfahrungsberichte, Rezensionen und Tests zu Tage zu fördern.

Empfehlungen

Der Übergang von Bewertungen zu tatsächlichen Empfehlungen ist fließend. Im Internet finden sich zu den verschiedensten Plugins ausführliche Tests und Erklärungen, die oft weiter über einfache Bewertungen hinausgehen. Mit etwas Glück sollten Sie so in den meisten Fällen abschätzen können, ob ein Plugin für Ihren Anwendungsfall tauglich ist.

Auf seelenlose Listen-Artikel wie „Diese 9 Plugins benötigt jede Website“ oder „Die 20 Game-Changer WordPress-Plugins 2021“ können Sie dabei im Übrigen getrost verzichten. Ein nennenswerter Erkenntnisgewinn dürfte hier in den meisten Fällen nicht stattfinden.

Wirklich geädelt wird ein Plugin aber erst durch eine persönliche Empfehlung. Wenn ein WordPress-Profi, dem Sie vertrauen, ein Plugin empfiehlt, sollte das in Ihrer Bewertung mindestens einen Bonuspunkt wert sein.

Sie finden keine derartigen Empfehlungen? Wenn Sie langjährige WordPress-Anwender*innen kennen, gelegentlich auf einem lokalen WordPress-Meetup vorbeischaun oder in Artikeln, Podcasts oder eBooks auf WordPress-Profis aufmerksam geworden sind, denen Sie das nötige Wissen zutrauen: fragen Sie einfach.

Die Entscheidung

Nachdem Sie einen oder mehrere Plugin-Kandidaten anhand dieser Punkte überprüft haben, müssen Sie wohl oder übel zu einer Entscheidung kommen. Einzelne Punkte können für Ihren individuellen Fall dabei durchaus größeres Gewicht haben als andere. Vielleicht erweitern Ihre Anforderungen meine exemplarische Checkliste aber auch noch. Zum Beispiel, weil Barrierefreiheit für Sie und Ihr Projekt besonders wichtig ist und ein Plugin, das hier punktet, automatisch die Nase vorn hat.

Wenn Sie am Ende die Wahl zwischen zwei vermeintlich gleich guten Plugins haben, testen Sie beide. In den allermeisten Fällen wird sich der Umgang mit einem der beiden besser anfühlen. Vor allem, wenn Sie viel und regelmäßig mit einem Plugin arbeiten, ist das abschließend ein Aspekt, den Sie nicht vernachlässigen sollten.



Der schnellste Weg zur sicheren WordPress-Website

- Direkt loslegen – WordPress vorinstalliert
- Regelmäßige System- & Sicherheit-Updates
- Kostenloses SSL-Zertifikate inkl.
- 1-Klick Backup und Wiederherstellung

So einfach geht's

www.df.eu

Domain **Factory**