



# WordPress für Fortgeschrittene

Einrichten, anpassen und entwickeln



**STRATO AG**

Pascalstraße 10  
10587 Berlin  
Berlin 2017  
[www.strato.de](http://www.strato.de)

**Autoren**

Christian Lingnau  
Franz Neumeier

**Redaktion**

Lisa Kopelmann  
Michael Poguntke  
Oliver Meiners

**Satz**

Katrin Krause

# Inhaltsverzeichnis

Einleitung	4
<b>1 Voraussetzungen</b>	<b>5</b>
1.1 WordPress bei STRATO manuell installieren	5
1.2 So nimmst Du Deinen Blog zu STRATO mit	10
1.3 Lokale Testumgebung erstellen	17
<b>2 Layout</b>	<b>23</b>
2.1 Child-Themes: Individuelles WordPress-Design ganz einfach	23
2.2 6 Schönheitstipps für das WordPress-Theme Twenty Nineteen	31
2.3 6 Schönheitstipps für den Kommentarbereich	37
2.4 Autor-Template erstellen	45
2.5 Benutzerdefinierte Felder anlegen	50
2.6 404-Fehlerseite individuell gestalten	55
2.7 Landing Page mit einem Seiten-Template erstellen	61
2.8 Eigenes Theme entwickeln: Bootstrap-Navigation in WordPress implementieren	68
2.9 Eigenes Theme entwickeln: Icons integrieren und Kommentarbereich optimieren	74
2.10 Eigenes Theme entwickeln: Strukturieren mit HTML und CSS	84
<b>3 Plugins</b>	<b>91</b>
3.1 Dein eigenes WordPress-Plugin leicht gemacht	91
3.2 Plugins ersetzen durch Code am Beispiel »Related Posts«	96
<b>4 Backend</b>	<b>101</b>
4.1 WordPress-Login bunter machen	101
4.2 Backend anpassen: Individuell statt von der Stange	105
<b>5 Sicherheit</b>	<b>110</b>
5.1 Strategischer Hintergrund: Welche Sicherheitslücken solltest Du schließen?	110
5.2 Potenzielle Sicherheitslücken von WordPress schließen	114
5.3 Typische WordPress-Fehler beheben	118
5.4 Statische Websites mit WordPress erstellen	127
5.5 Plugins als Sicherheitsrisiko	131
<b>6 Performance</b>	<b>132</b>
6.1 WordPress Beine machen per .htaccess	132
<b>7 Tracking</b>	<b>135</b>
7.1 An diesen Zahlen erkennst Du, ob Dein Blog erfolgreich ist	135
7.2 Content-Gruppen: Traffic detailliert aufschlüsseln mit WordPress und Google Analytics	143
<b>8 WordPress Multisite</b>	<b>154</b>
8.1 Multisite mit Verzeichnissen und Subdomains	154
8.2 Domain Mapping mit WordPress Multisite	158
Schlusswort	161

# Einleitung

## Liebe Bloggerin, lieber Blogger,

ein eigener Reise-Blog, ein Unternehmens-Blog oder ein Online-Auftritt für das ganz private Projekt – mit WordPress alles kein Problem.

Vielleicht kennst Du schon unser E-Book [›WordPress für Einsteiger](#) und hast bereits eine eigene Online-Präsenz mit WordPress erstellt. Umso besser! Unser zweites E-Book »WordPress für Fortgeschrittene« zeigt Dir, wie Du noch mehr aus Deinem Blog herausholen kannst.

**Du bist noch nicht mit einem passenden Paket ausgestattet?**

[›Wähle hier den WordPress-Tarif, der zu Deinen Anforderungen passt.](#)

Unsere Autoren Christian Lingnau und Franz Neumeier geben Dir Experten-Tipps dazu an die Hand, wie Du mit Deinem WordPress startest, Deinem Blog ein einzigartiges Aussehen verleihst, eigene Plugins erstellst und typische Fehler behebst. Sie erklären Dir außerdem, welche Sicherheitslücken Du beachten solltest, wie Du sie beheben kannst und woran Du erkennst, ob Dein Blog erfolgreich ist.

Alle Texte und Beispiele basieren auf unserem [›Hosting-Paket WordPress Plus](#). Kurz nachdem dieses E-Book fertiggestellt war, kam das neue WordPress 5.0. Wir haben entsprechend das Kapitel 2.2 für die neue Version angepasst. Gleichzeitig bitten wir um Verständnis, dass sich manche Screenshots noch auf WordPress 4.9 beziehen.

Wir wünschen Dir viel Spaß beim Lesen und Bloggen!

Dein STRATO Team

# 1 Voraussetzungen

## 1.1 WordPress bei STRATO manuell installieren

WordPress installierst Du bei STRATO Hosting-Paketen über den AppWizard mit einem Mausklick. Willst Du mehr Kontrolle über die WordPress-Installation, richtest Du WordPress mit dieser Anleitung unkompliziert von Hand ein.

Die manuelle Installation von WordPress verläuft in drei Schritten: Datenbank anlegen, WordPress vorbereiten sowie die Installation im Webspaces inklusive der Anpassung einiger Optionen durchführen.

### 1. Datenbank anlegen

Sofern Du in Deinem Hosting-Account über genügend Datenbanken verfügst, empfiehlt es sich, für WordPress eine eigene Datenbank zu verwenden, statt sie mit anderen Anwendungen zu teilen. Du kannst aber auch mit einer bereits vorhandenen Datenbank arbeiten, denn WordPress nutzt für die Tabellen standardmäßig das Präfix **wp\_** und vermeidet so Kollisionen mit anderen Anwendungen.

Um eine neue Datenbank anzulegen, klickst Du einfach auf den Button **Datenbank anlegen** in Deinem STRATO Kunden-Login unter **Datenbanken und Webspaces > Datenbankverwaltung**.

<input type="checkbox"/>	Datenbank	Benutzername	Belegt	Kommentar
<input type="checkbox"/>	...	...	23.5 MB	Datenbank erstellt am 26.03.2017
<input type="checkbox"/>	...	...	7.9 MB	AppWizard: Datenbank für TYPO3 3 1/ 2018

Die Informationen zur neu angelegten Datenbank werden nun angezeigt. Notiere Dir Datenbank- und Benutzernamen für die spätere WordPress-Installation.

## 2. Datenbank-Passwort ändern

Ändere nun das Passwort der Datenbank und verwende dabei eine möglichst lange Kette aus Groß- und Kleinbuchstaben sowie Ziffern. Notiere das neue Passwort ebenfalls für später und leg es gut ab. Du benötigst es später noch mal. Am besten schreibst Du das Passwort in einem Text-Editor und kopierst es in das Formular. So vermeidest Du Tippfehler auch bei komplizierten – und damit sicheren – [Passwörtern](#).

**Datenbankverwaltung**

Datenbankname

Unsere Empfehlung für gute Passwörter

Neues Passwort

••••••••••

"sehr gut"

Zurück Passwort ändern

**Anforderung**  
 Mindestens 10 Zeichen  
 Maximal 128 Zeichen  
 Erlaubte Zeichen:  
 Buchstaben: a-z und A-Z, keine Umlaute  
 Ziffern: 0-9  
 Sonderzeichen: !#\$%&()\*+,-./:;  
 <>=?@[!^\_{|}~

In der Übersicht kannst Du auch den Kommentar zur Datenbank ändern, beispielsweise in »wordpress«, damit Du später weißt, welche Datenbank zu Deiner WordPress-Installation gehört.

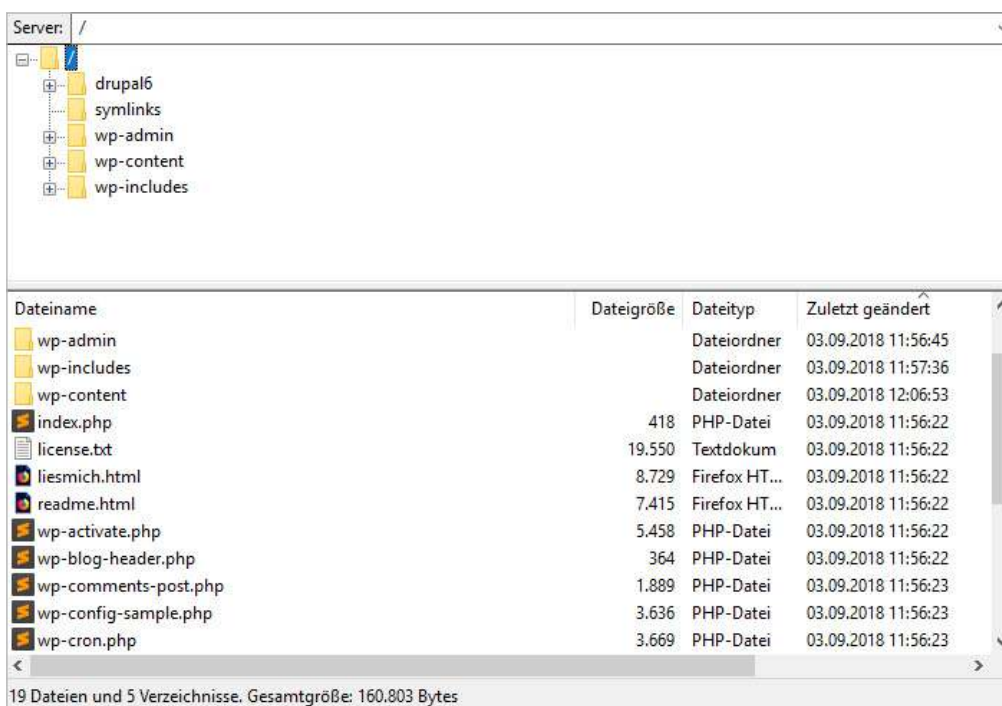
## 3. WordPress herunterladen

Jetzt lädst Du die aktuellste Version von WordPress herunter:

› <https://de.wordpress.org/download/>. Entpacke die heruntergeladene ZIP-Datei auf Deiner lokalen Festplatte. Kontrolliere, ob dabei die Verzeichnisstruktur des Unterordners »wordpress« komplett erhalten geblieben ist. Sie enthält die Verzeichnisse **wp-admin**, **wp-includes** und **wp-content** sowie einige PHP-Dateien.

## 4. WordPress hochladen

Lade nun den Inhalt des Verzeichnisses »wordpress« in Deinen Weospace. Am besten verwendest Du dazu den [SFTP-Zugang](#). Die Verzeichnisstruktur sieht danach so aus:



## 5. WordPress installieren

Wenn Du nun das WordPress-Unterverzeichnis Deiner Domain aufrufst (z. B. <http://katzenfreunde-suedsachsen.de/wordpress>), startet die Installationsroutine. Gib im zweiten Dialog den zuvor notierten Datenbanknamen, Benutzernamen und das Passwort ein. Als Datenbank-Host trägt Du »rdbms.strato.de« (statt »localhost«) ein.

Ändere außerdem das standardmäßige Tabellen-Präfix: Statt **wp\_** könntest Du beispielsweise Deine Initialen oder eine andere, kurze und willkürliche Buchstabenfolge verwenden. Das bietet zusätzlichen Schutz vor Hackern.

Hier sollten die Zugangsdaten zu deiner Datenbank eingetragen werden. Im Zweifel frage bitte beim Support deines Webhostings nach.

<b>Datenbank-Name</b>	<input type="text"/>	Der Name der Datenbank, die du für WordPress verwenden möchtest.
<b>Benutzername</b>	<input type="text"/>	Dein Datenbank-Benutzername.
<b>Passwort</b>	<input type="password"/>	Dein Datenbank-Passwort.
<b>Datenbank-Host</b>	<input type="text" value="rdbms.strato.de"/>	Sollte localhost nicht funktionieren, erfrage bitte den korrekten Wert beim Support deines Webhostings.
<b>Tabellen-Präfix</b>	<input type="text" value="wp_"/>	Falls du mehrere WordPress-Installationen innerhalb einer Datenbank aufbauen möchtest, ändere diesen Eintrag.

Sobald die Verbindung zur Datenbank hergestellt ist, musst Du noch ein paar grundlegende Daten eingeben, bevor die eigentliche Installation startet: den Namen Deines Blogs, einen Benutzernamen und ein Passwort (beides notieren!) sowie Deine E-Mail-Adresse. Der Benutzername, den Du hier anlegst, bekommt Administrations-Rechte. Wähle daher einen Benutzernamen, der nicht ohne Weiteres für Dritte erkennbar ist und keinesfalls »admin« oder »Administrator« lautet.

Unter »Sichtbarkeit für Suchmaschinen« entscheidest Du, ob Suchmaschinen Dein Blog indexieren dürfen. Obwohl Du das in den meisten Fällen natürlich willst, solltest Du hier zunächst ein Häkchen setzen und Suchmaschinen so davon abhalten, Deine Website zu listen. Erst nachdem Du Deinen Blog fertig eingerichtet hast (u. a. Impressum und Datenschutz-Hinweise), solltest Du das Indexieren erlauben.

Denke daran, das Häkchen später wieder zu entfernen, denn sonst sperrst Du Google und andere Suchmaschinen dauerhaft aus. Um diese Option zu ändern, gehst Du in der WordPress-Admin-Oberfläche in den Bereich **Einstellungen > Lesen**.



## Willkommen

Willkommen bei der berühmten 5-Minuten-Installation von WordPress! Gib unten einfach die benötigten Informationen ein und schon kannst du starten mit der am besten erweiterbaren und leistungsstarken persönlichen Veröffentlichungsplattform der Welt.

## Benötigte Informationen

Bitte trage die folgenden Informationen ein. Keine Sorge, du kannst all diese Einstellungen später auch wieder ändern.

**Titel der Website**

**Benutzername**

Benutzernamen dürfen nur alphanumerische Zeichen, Leerzeichen, Unterstriche, Bindestriche, Punkte und das @-Zeichen enthalten.

**Passwort**

Stark

**Wichtig:** Du wirst dieses Passwort zum Anmelden brauchen. Bitte bewahre es an einem sicheren Ort auf.

**Deine E-Mail-Adresse**

Bitte überprüfe nochmal deine E-Mail-Adresse auf Richtigkeit, bevor du weitermachst.

**Sichtbarkeit für Suchmaschinen**

Suchmaschinen davon abhalten, diese Website zu indexieren.

Es ist Sache der Suchmaschinen, dieser Bitte nachzukommen.

Abschließend klickst Du auf **WordPress installieren**. Danach kannst Du Dich mit Deinen zuvor angelegten Zugangsdaten einloggen.

## 1.2 So nimmst Du Deinen Blog zu STRATO mit

Einen WordPress-Blog von einem anderen Web-Hoster umzuziehen, ist mithilfe des Plugins ›[Duplicator](#) bequem zu bewältigen. Mit ein paar Tricks und einer Schritt-für-Schritt-Anleitung gelingt Dir der Umzug ganz leicht.

Der Umzug eines WordPress-Blogs ist eigentlich keine große Sache: Datenbank und Dateien kopieren, ein paar Konfigurations-Dateien anpassen, fertig. Aber wie so oft steckt der Teufel im Detail. Mit dem Plugin Duplicator gehst Du den meisten potenziellen Problemen aus dem Weg. Das Plugin kümmert sich automatisch um die richtige Konfiguration und prüft vorab, ob die nötigen Voraussetzungen für einen Umzug wie richtige PHP-Versionen und Ähnliches gegeben sind.

Damit der Wechsel zu STRATO schnell gelingt, bieten wir eine ›[Schritt-für-Schritt-Anleitung](#) an.

Falls Du mehrere WordPress-Instanzen installieren möchtest, solltest Du von Anfang an Unterverzeichnisse anlegen. Daneben sind für einen WordPress-Umzug aber noch einige weitere Überlegungen sinnvoll, die wir in den folgenden Tipps zusammenfassen.


### 1. WordPress zusammen mit der Domain umziehen


Wenn Du einen bereits vorhandenen WordPress-Blog umziehen willst, müsste die Domain theoretisch auf zwei Webservern gleichzeitig erreichbar sein. Denn Deine Domain zeigt zunächst auf den bisherigen Webspace, der Blog muss beim Umzug zum neuen Server aber möglichst schnell wieder unter dieser Domain erreichbar sein. Wichtig für Dich zu wissen: In manchen Fällen kann es bis zu 48 Stunden dauern, bis alle DNS-Server die vorgenommenen Änderungen übernommen haben – darauf hast Du leider keinen Einfluss.


Entscheidend ist die Reihenfolge der einzelnen Schritte:



- Installiere in Deinem alten WordPress das Plugin Duplicator und lege wie in der Anleitung beschrieben ein Backup an. Lade dann das Archiv und den Installer auf Deinen Computer herunter.

### Erstellungs-Status

 **Archiv komplett**  
Verarbeitungszeit: 6.87 sec.

Dateien herunterladen 

 **Installer**     **Archiv** (10.69MB)

  **Ein-Klick-Download** 

[Wie installiere ich dieses Paket?](#)

---

*[Lief mir, das Plugin zu bewerten!](#)*

- Lade das .zip-Archiv und die **installer.php** probeweise wie in der Anleitung beschrieben in den neuen Webpace. Welche Domain Du dafür verwendest, ist egal. Es handelt sich lediglich um eine Testinstallation, die Du danach wieder löschst.
- Die Installationsroutine startet, sobald Du den Installer über die neue Domain aufrufst. Den Pfad im Bereich **New Settings** musst Du manuell anpassen, da der automatisch vom Plugin ermittelte Pfad später zu Problemen führen wird. Den korrekten Webpace-Pfad (z. B. /mnt/rid/56/78/12345678/htdocs/) findest Du im STRATO Kundenlogin unter **Ihr Paket > Übersicht**. Falls Du WordPress in einem Unterverzeichnis installieren möchtest, gibst Du dieses direkt dahinter an.

## Duplicator

[version: 1.2.52](#)  
[help » unlocked](#)

Mode: Standard  
[dup-installer-log.txt](#)

### Step 3 of 4: Update Data

#### New Settings

**URL:**  [get](#)

**Path:**

**Title:**

#### Options

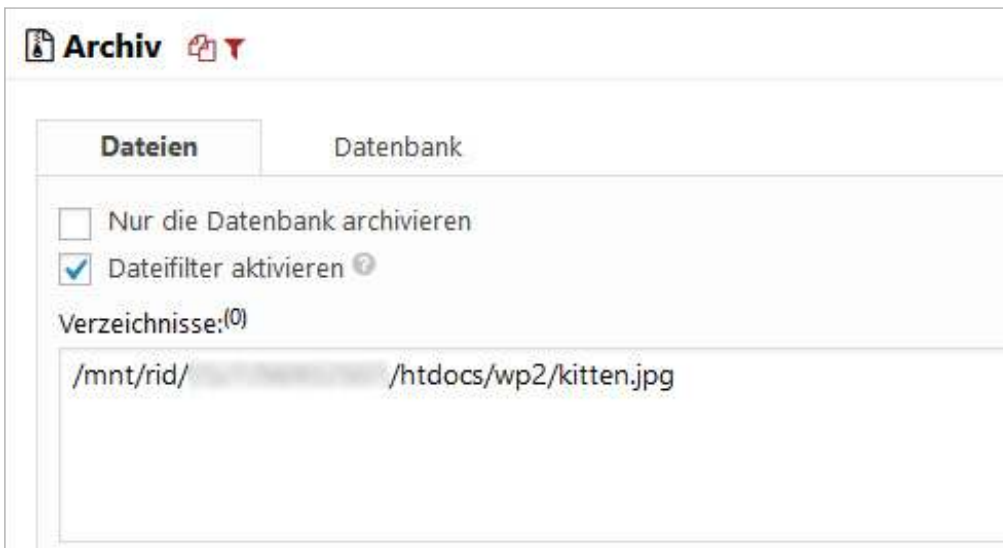
- Erst wenn Du auf diese Weise getestet hast, dass alles funktioniert, ziehst Du Deine Domain vom alten Provider zu STRATO um.
- Sobald die Domain bei STRATO erreichbar ist, spielst Du nun das Duplicator-Archiv endgültig ein und verwendest dafür Deine eigentliche Domain.

Beachte, dass Dein Blog auf dem alten Server nicht mehr erreichbar ist, sobald die Domain umgezogen ist. Je schneller Du dann das Archiv im neuen Webpace einspielst, desto kürzer ist Dein Blog offline.

## 2. Bilder und große Dateien manuell umziehen

Eine Stärke des Duplicator-Plugins ist der Umzug von Datenbank und Dateien in einem Rutsch. Bei kleinen Blog-Projekten funktioniert das bestens. Sobald Du aber eine größere Zahl an Bildern in WordPress hochgeladen hast oder andere große Dateien wie MP3- oder Video-Dateien vorhanden sind, empfiehlt es sich, diese separat umzuziehen.

Die Anleitung beschreibt ausführlich, wie das geht: Erstellst Du das Archiv, klammerst Du bestimmte Verzeichnisse aus und überträgst die Dateien dann per SFTP. Achte darauf, dabei aber die Ordnerstruktur nicht zu verändern, denn sonst findet WordPress die Dateien nicht mehr.



**Archiv**

**Dateien** Datenbank

Nur die Datenbank archivieren

Dateifilter aktivieren

Verzeichnisse: (0)

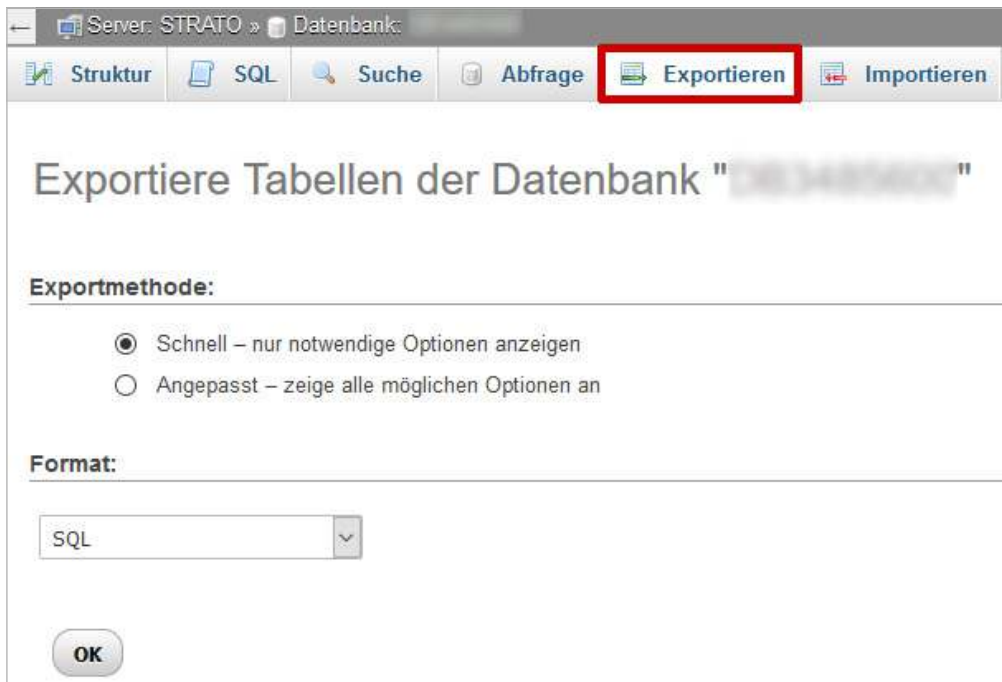
/mnt/rid/ /htdocs/wp2/kitten.jpg

Wenn Du große Dateien auf diese Weise umziehst, statt sie in das Duplicator-Archiv aufzunehmen, erledigt das Plugin seine Aufgabe deutlich schneller und das Fehlerrisiko beim Schreiben der Archiv-Datei sinkt.

### 3. Auf Nummer sicher gehen

Auch wenn Du die alte WordPress-Installation vermutlich nicht mehr brauchen wirst: Lege vor dem Umzug ein Backup des gesamten Webspace (per SFTP) und der Datenbank an. Damit bist Du auf der sicheren Seite, wenn beim Umzug etwas schief laufen sollte. Dann kannst Du zumindest den Status vor dem Umzug schnell wiederherstellen.

Tipp: Mit dem Tool [phpMyAdmin](#) ist ein Datenbank-Backup besonders schnell erledigt. Die Größe der Sicherungsdatei ist auf 32 MB begrenzt. Größere Backups kannst Du per SSH anlegen.

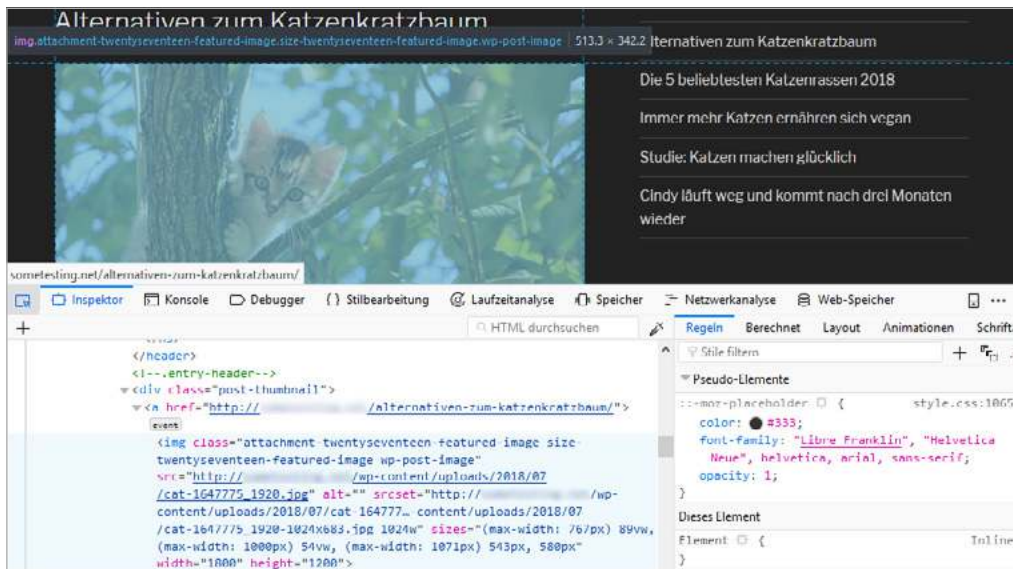


Die ursprüngliche WordPress-Installation im bisherigen Webspac sollst Du erst löschen, wenn Du absolut sicher bist, dass der Umzug funktioniert hat. Kleinere Probleme, etwa einzelne fehlende Bilder, findest Du möglicherweise erst später. Denn irgendetwas übersieht man fast immer. Ist die alte WordPress-Installation noch vorhanden, kannst Du gezielt nach Unterschieden zwischen der alten und neuen Installation suchen, um Fehlerquellen auf die Spur zu kommen.

#### 4. Und wenn es dennoch einmal hakt ...

Jeder Webserver ist ein wenig anders konfiguriert, WordPress-Themes nutzen unterschiedliche PHP-Funktionen – das kann zu Problemen beim WordPress-Umzug führen. Was kannst Du also tun, wenn nach dem Umzug irgendetwas nicht funktioniert, Bilder fehlen oder Ähnliches?

Um die möglichen Fehler einzugrenzen, solltest Du Dir zunächst den HTML-Quellcode im Browser ansehen und prüfen, ob dort beispielsweise falsche Verzeichnispfade vorkommen. Webbrowser haben meist eine eingebaute Entwickler-Funktion, über die Du den Quellcode analysieren kannst. In Firefox und Chrome zum Beispiel rufst Du diese über die Taste F12 auf.



Entwicklerwerkzeuge in Firefox

Hast Du beispielsweise einen falschen Verzeichnispfad gefunden, kannst Du das über ein Suchen-und-Ersetzen-Plugin korrigieren (siehe [5 kleine Helferlein, die das Blogger-Leben leichter machen](#)). Liegt der Ursprung des Problems in einer der Theme-Dateien, lädst Du das komplette Theme-Verzeichnis per SFTP auf Deinen Computer und benutzt die dateiübergreifende Suchfunktion Deines Editors (z. B. [Notepad++](#)). Einmal gefunden, kannst Du den Fehler direkt in der Theme-Datei korrigieren.

## 5. Dem Google-Ranking zuliebe nicht zu viel gleichzeitig ändern

Die Versuchung ist groß, bei einem Server-Umzug auch all die Dinge zu verändern, die man eh schon lange einmal anpacken wollte. Doch genau das solltest Du vermeiden. Behalte auf jeden Fall erst einmal Deine Domain bei, übernimm die bisherige URL-Struktur und verändere das Theme nicht. Das hat zwei handfeste Vorteile:

1. Je weniger Du veränderst, desto weniger potenzielle Fehlerquellen gibt es. Gerade bei mehreren, vermeintlich voneinander unabhängigen Veränderungen entstehen oft Wechselwirkungen, die nur schwer zu identifizieren sind.
2. Suchmaschinen reagieren empfindlich auf globale Änderungen von Websites. Durch den Umzug verändert sich ohnehin schon die IP-Adresse Deines Blogs. Das ist für Google ein Signal, für eine Weile genauer hinzusehen, ob die Domain

womöglich verkauft wurde und nun vielleicht als Spam-Schleuder dient. Würdest Du jetzt auch noch das Design – und damit die HTML-Struktur – verändern, wäre das für Google ein weiteres Alarm-Signal. Wählst Du eine neue Permalink-Struktur, verändern sich die URLs, was ebenfalls negative SEO-Folgen haben kann.

War Dein Umzug erfolgreich, kannst Du nach ein paar Wochen die nächsten Veränderungen in Angriff nehmen. Aber auch hier gilt: Lieber in kleinen Schritten, um keine Probleme mit Suchmaschinen zu riskieren.

## 6. Geduld und gute Vorbereitung vermeiden Ärger

Ein Tipp zum Schluss: Auch wenn der Umzug von WordPress verhältnismäßig einfach ist und in den meisten Fällen glattläuft, bleibt doch immer ein Restrisiko. Um das so gering wie möglich zu halten, solltest Du sehr konzentriert arbeiten und geduldig sein. Wirft die Prüfung des Duplicator-Plugins eine Warnung aus oder sind Dir Details beim Ablauf unklar, recherchiere und teste vorab so lange, bis alle Zweifel ausgeräumt sind. Die eigene Ungeduld verleitet oft zur Unachtsamkeit. Bedenke daher: Geduld spart gerade bei einer Aktion wie dem Umzug eines WordPress-Blogs eine Menge Ärger.



## 1.3 Lokale Testumgebung erstellen

Neue Funktionen, Plugins oder Designs kannst Du entweder direkt online oder in einer lokalen Testumgebung testen. Letzteres ist sicherer, weil Du unabhängig von der Website arbeitest und Dir keine Gedanken über Sicherheitsrisiken machen musst. Wir zeigen Dir im Folgenden, wie Du eine einfache lokale Testumgebung für WordPress erstellst.

Wer den »[White Screen of Death](#)« kennt, kann ein Lied davon singen: Themes und Plugins live auszuprobieren, ist keine gute Idee. Selbst gut gepflegte Software kann im Zusammenspiel mit anderen Programmen Probleme verursachen. Riskant sind vor allem Codeanpassungen im laufenden Betrieb. Und wer ein eigenes Theme entwickeln möchte, macht dies ohnehin außerhalb der Produktivumgebung.

Folglich solltest Du Anpassungen immer vorab (offline) testen. Für die Entwicklung eines eigenen Themes sind das Starter-Theme Underscores und das Framework Bootstrap 4 eine ideale Basis.

### Schritt 1: InstantWP starten und WordPress aktualisieren

Das ideale Werkzeug für unser Vorhaben ist »[InstantWP](#)«: Das Programm simuliert den benötigten Server und enthält ein unter Windows und MacOS lauffähiges WordPress. Dazu musst Du nur die .zip-Datei herunterladen und die Datei Start-InstantWP doppelklicken. Vielleicht meckert Dein Antiviren-Programm und Du musst den Zugriff durch die Firewall zulassen – doch keine Sorge, InstantWP ist Open Source Software und frei von Viren oder Trojanern.

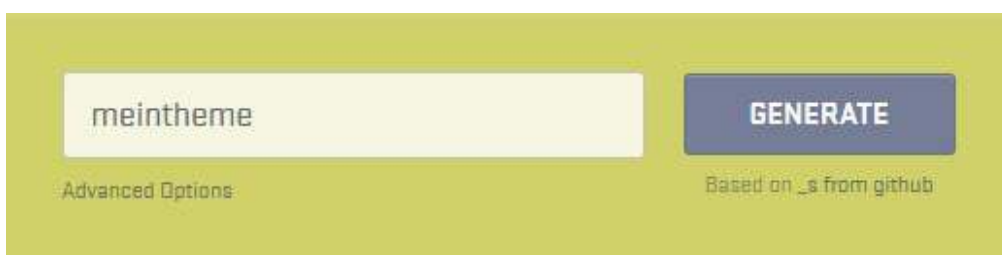
Nach dem Start kannst Du Dich über den Administrationsbereich von InstantWP (**WordPress Admin**) in das WordPress-Backend einloggen (Benutzer: »admin«, Passwort: »password«). Als Erstes solltest Du WordPress aktualisieren.



Administrationsbereich von InstantWP

## Schritt 2: Theme installieren

Um Anpassungen an Deiner bestehenden Website vorab zu testen, installierst Du auf Deiner Testumgebung das gleiche Theme und übernimmst dieselben Einstellungen Deiner Website. Ist die Testumgebung für die Entwicklung eines eigenen Themes gedacht, nutzt Du am besten das Basis-Theme »[Underscores](#)«. Damit hast Du schon mal ein sauber programmiertes Fundament für Dein eigenes Layout. Gib dem Theme einen Namen (z. B. »meintHEME«), klicke auf **Generate** und lade dann die .zip-Datei herunter.



Basis-Theme per Knopfdruck: Underscores

Nachdem Du das Theme installiert und aktiviert hast, kannst Du Dir das Ergebnis über das InstantWP Control Panel (**WordPress Frontpage**) oder wie gewohnt über das WordPress-Backend (**Zur Website**) anschauen:


**Meine Spielwiese**

Lokale Testumgebung für WordPress

[Sample Page](#)

**Lorem ipsum dolor sit amet**

Posted on [August 25, 2017](#) by [admin](#)



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Posted in [Uncategorized](#) [Leave a Comment](#) [Edit](#)

**Hello world!**

Posted on [Januar 10, 2017](#) by [admin](#)

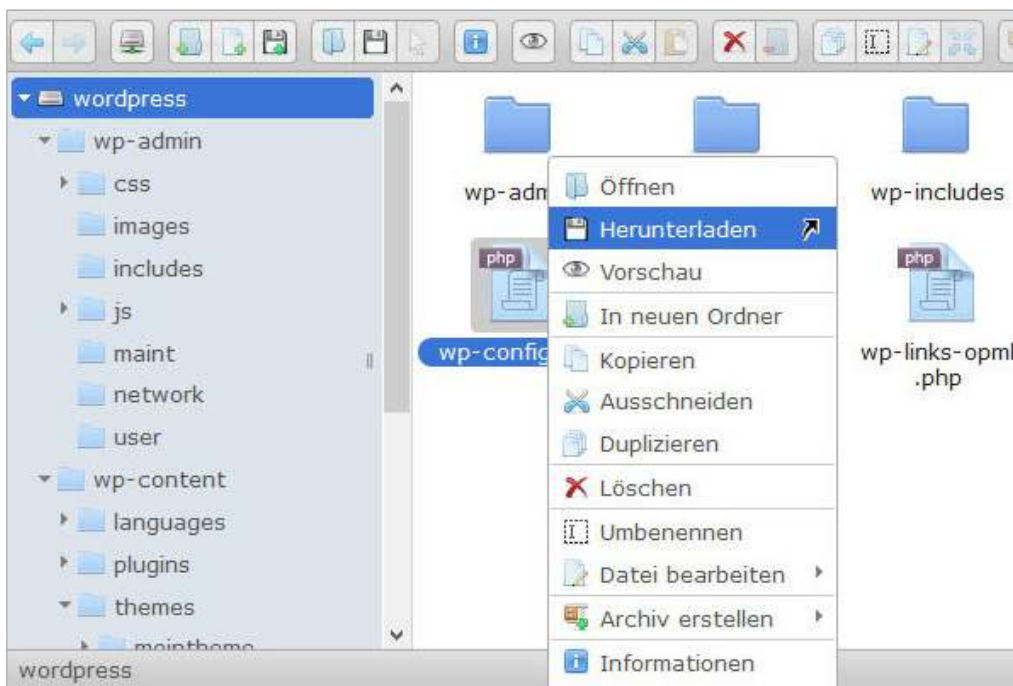
Das Layout von Underscores ist im Originalzustand bewusst minimalistisch. Damit bleibt mehr Raum für individuelle Anpassungen.

Aus Sicherheitsgründen sollte mindestens ein weiteres Theme installiert sein, auf welches WordPress im Notfall zurückgreifen kann.

### Schritt 3: Debug-Modus aktivieren

Zu einer Testumgebung gehört auch das sogenannte Debugging, also das Aufspüren und Beheben von Bugs. Damit WordPress bei fehlerhaftem PHP-Code statt einer weißen Seite eine aussagekräftige Fehlermeldung ausgibt, musst Du den Debug-

Modus aktivieren. Rufe dazu den Web File Manager von InstantWP auf (**Control Panel > Themes Folder**) und navigiere in das WordPress-Hauptverzeichnis.



Der InstantWP Web File Manager funktioniert wie ein FTP-Programm – nur offline.

Per Rechtsklick lädst Du die Datei **wp-config.php** herunter und ersetzt die Zeile

```
define('WP_DEBUG', false);
```

durch diese:

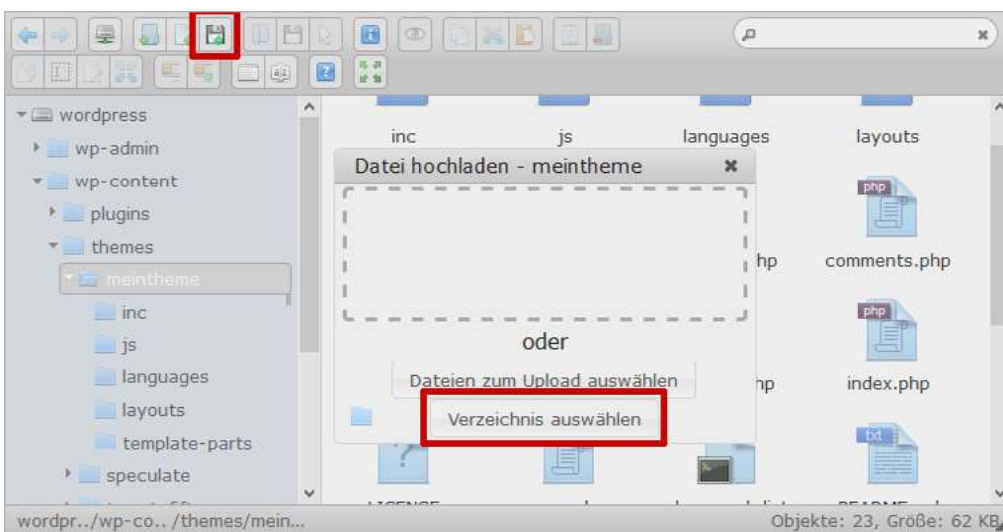
```
define('WP_DEBUG', true);
```

Denke daran, diese Einstellung wieder auf **false** zurückzusetzen, bevor Du das fertige Theme in Deinem Blog online nimmst.

## Schritt 4: Bootstrap einbinden

Bootstrap ist ein beliebtes Open Source Framework mit fertigen Buttons, Navigationsleisten, Grid-Systemen, Tabellen und vielem mehr. Als Basis dienen CSS-Komponenten, Schriften, Icons und JavaScript-Code. Um den Baukasten nutzen zu können, musst Du Bootstrap mit WordPress verbinden.

Dazu lädst Du [Bootstrap in der aktuellen Version \(4.1.1\)](#) herunter (Compiled CSS and JS). Entpacke die Datei und benenne den Ordner in »bootstrap« um. Im InstantWP Control Panel wählst Du den Bereich **Themes Folder** und lädst den Bootstrap-Ordner in das Theme-Verzeichnis (`wordpress/wp-admin/themes/meintheme`):



Um WordPress und Bootstrap miteinander zu verknüpfen, musst Du jetzt nur noch die `functions.php` Deines Themes anpassen.

**Hinweis:** Die nachfolgenden Zeilenangaben für den Code können von denen Deiner Underscores-Version abweichen.

Um Zeile 122 befindet sich `function meintheme_scripts()` – statt »meintheme« steht dort der Name Deines Themes. Darunter folgen mehrere Funktionen. Füge vor der schließenden geschweiften Klammer `}` die folgenden Zeilen ein. Achte dabei darauf, im Code die jeweils gültige Bootstrap-Version anzugeben (in diesem Beispiel 4.1.1):

```
// Bootstrap
wp_register_script(,bootstrap-js', get_template_directory_uri() .
./bootstrap/js/bootstrap.min.js', array(,jquery'), ,4.1.1', true );
wp_register_style(,bootstrap-css', get_template_directory_uri() .
./bootstrap/css/bootstrap.min.css', array(), ,4.1.1', ,all' );
wp_enqueue_script(,bootstrap-js');
wp_enqueue_style(,bootstrap-css');
```

Im Editor sieht das so aus:

```

/**
 * Enqueue scripts and styles.
 */
function meintheme_scripts() {
    wp_enqueue_style( 'meintheme-style', get_stylesheet_uri() );

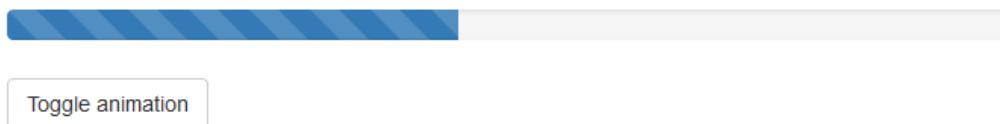
    wp_enqueue_script( 'meintheme-navigation', get_template_directory_uri() . '/js/navigation.js', array(), '20151215', true );
    wp_enqueue_script( 'meintheme-skip-link-focus-fix', get_template_directory_uri() . '/js/skip-link-focus-fix.js', array(), '20151215', true );

    if ( ! is_singular() && comments_open() && get_option( 'thread_comments' ) ) {
        wp_enqueue_script( 'comment-reply' );
    }

    // Bootstrap
    wp_register_script( 'bootstrap-js', get_template_directory_uri() . '/bootstrap/js/bootstrap.min.js', array( 'jquery' ), '4.1.1', true );
    wp_register_style( 'bootstrap-css', get_template_directory_uri() . '/bootstrap/css/bootstrap.min.css', array(), '4.1.1', 'all' );
    wp_enqueue_script( 'bootstrap-js' );
    wp_enqueue_style( 'bootstrap-css' );
}
add_action( 'wp_enqueue_scripts', 'meintheme_scripts' );

```

Nach dem Upload ist das Theme mit Bootstrap verknüpft. Damit kannst Du Bootstrap-Elemente in Deinem Blog testen: Kopiere einfach die vorgefertigten Code-Blöcke von [getbootstrap.com](https://getbootstrap.com) und füge sie in den WordPress-Editor (Text-Ansicht) ein. Nach dem Veröffentlichen erscheinen die Elemente im Frontend Deines Blogs.



Zu den Bootstrap-Komponenten gehören zum Beispiel animierte Fortschrittsbalken.

## Spielwiese für WordPress

Ob Theme-Entwicklung oder -Anpassungen: Mit den vier gezeigten Schritten kannst Du sämtliche Änderungen Deiner privaten Website oder Deines Blogs bequem und gefahrlos vorab testen. Achte darauf, dass Du identische Einstellungen wählst und WordPress, Theme und Plugins auf dem gleichen aktuellen Stand sind. Wenn alles reibungslos funktioniert, lädst Du die Dateien hoch. Im nächsten Kapitel erfährst Du, wie Du das Layout flexibel weiterentwickelst.

# 2 Layout

## 2.1 Child-Themes: Individuelles WordPress-Design ganz einfach

Fertige Website-Vorlagen – Themes – gibt es massenhaft. Doch selten findet sich eines, das exakt passt. Fast immer sind kleine Anpassungen nötig – und sei es nur die Schriftart, ein paar Farben oder eine zu dick geratene Linie. Schon mit HTML- und CSS-Grundkenntnissen kannst Du WordPress-Themes ganz einfach verändern und anpassen. Für größere Änderungen ist Basis-Wissen in PHP hilfreich.

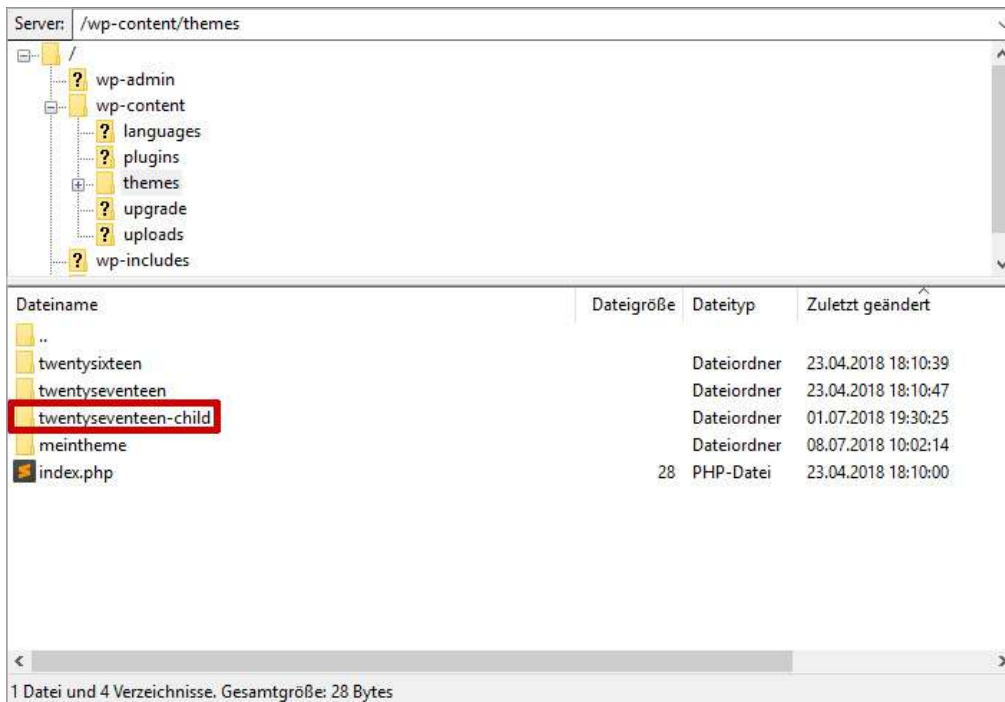
Mit sogenannten »Child-Themes« bietet WordPress ein mächtiges Feature. Sie setzen auf ein vorhandenes Theme mit dessen Formatierungen und Funktionen auf und ergänzen oder modifizieren es mit eigenen Elementen und Designs. Der große Vorteil: Das Original-Theme bleibt dabei unangetastet, Updates des Originals beeinträchtigen die Anpassungen im Child-Theme nicht.

### Schritt für Schritt: Erstelle Dein eigenes Child-Theme

Ein wenig Aufwand bringt nur das erstmalige Anlegen des Child-Themes mit sich. Danach sind die Anpassungen genau so unkompliziert als würdest Du direkt im Original-Theme arbeiten. Zunächst suchst Du Dir ein Theme aus, das Deinen Vorstellungen so nahe wie möglich kommt, und installierst es. Achte darauf, dass es als geeignet für Child-Themes beschrieben wird. Die meisten Themes sind das, aber es gibt Ausnahmen. Im Folgenden beschreiben wir, wie Du ein Child-Theme manuell anlegst.

#### 1. Ordner fürs Child-Theme anlegen

Themes liegen in Deinem Webspaces im Verzeichnis **wp-content/themes**. Wir verwenden für unser Beispiel hier ein Parent-Theme, das im Verzeichnis **wp-content/themes/parenttheme** liegt. Entsprechend musst Du bei den folgenden Beschreibungen **parenttheme** jeweils gegen den Verzeichnisnamen Deines tatsächlichen Parent-Themes austauschen.



Der Child-Theme-Ordner liegt auf derselben Verzeichnis-Ebene wie das Parent-Theme.

Unter **wp-content/themes** legst Du nun also auf der gleichen Ebene wie das ausgewählte Parent-Theme ein neues Verzeichnis an. Wie Du das Verzeichnis benennst, ist egal. Es bietet sich aber zum späteren Verständnis beispielsweise **parenttheme-child** als Name an.

## 2. Child-Theme definieren

In diesem neuen Verzeichnis – im Beispiel **wp-content/themes/parenttheme-child** – legst Du eine neue Datei namens **style.css** an. Sie muss die folgenden Code-Zeilen beinhalten, damit WordPress erkennt, dass es sich hier um ein Child-Theme zum Theme im Verzeichnis **wp-content/themes/parenttheme** handelt:



```

/*
Theme Name: Dein Child-Theme
Description: Beschreibung Deines Child-Themes
Author: Dein Name
Author URI:
Template: parenttheme
Version: 1.0
Tags: parenttheme
Text Domain: parenttheme-child
*/

```

Entscheidend ist die Zeile **Template**, denn dort muss der exakte Name des Verzeichnisses eingetragen sein, in dem das ausgewählte Parent-Theme liegt. In unserem Beispiel also **parenttheme**.

```

1  /*
2  Theme Name: Dein Child Theme
3  Description: Beschreibung Deines Child Themes
4  Author: Dein Name
5  Author URI:
6  Template: parenttheme
7  Version: 1.0
8  Tags: parenttheme
9  Text Domain: parenttheme-child
10 */

```

Das Child-Theme funktioniert nur, wenn das Parent-Template korrekt benannt ist.

### 3. Stylesheets des Parent-Themes verknüpfen

Lege nun im Hauptverzeichnis Deines Child-Themes eine neue Datei namens **functions.php** an. Diese Datei brauchst Du, um die CSS-Definitionen aus dem Parent-Theme mit dem Child-Theme korrekt zu verknüpfen und später gegebenenfalls auch für PHP-Anpassungen des Themes. Füge folgende Zeilen in die **functions.php** ein. Damit legst Du fest, dass zunächst das Stylesheet des Parent-Themes und dann das des Child-Themes geladen wird:

```

<?php
add_action( ,wp_enqueue_scripts', ,theme_enqueue_styles' );
function theme_enqueue_styles() {
    wp_enqueue_style( ,parent-style', get_template_directory_uri() . ,/style.css' );
    wp_enqueue_style( ,child-style', get_stylesheet_directory_uri() . ,/style.css',
array(,parent-style' ) );
}

```

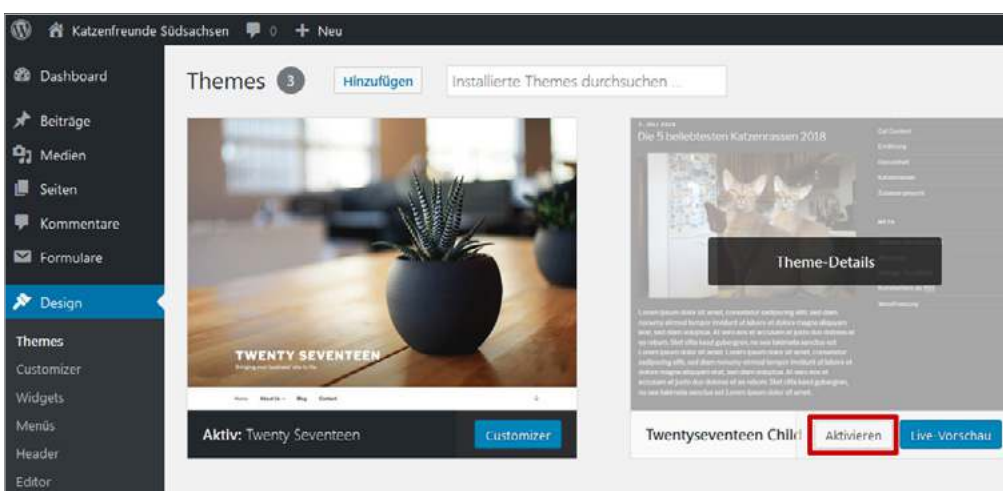
Übrigens: Die früher verwendete Methode, die CSS-Datei des Parent-Themes über @import in der style.css zu integrieren, wird von WordPress nicht mehr empfohlen. Sie hat vor allem Performance-Nachteile.

Falls diese Methode bei Deinem Theme nicht funktioniert, findest Du [hier eine alternative Anleitung](#).

#### 4. Child-Theme aktivieren

Bevor Du das das Theme aktivierst, kannst Du noch ein Vorschaubild für die Themes-Übersicht im Backend machen. Erstelle dazu einen Screenshot Deines Child-Themes und lege diesen in der Größe 880×660 Pixel mit dem Dateinamen **screenshot.jpg** ins Hauptverzeichnis Deines Child-Themes. WordPress erkennt das Vorschaubild automatisch und stellt Dir nun auch Dein Child-Theme in der Übersicht optisch ansprechend dar. Diesen Schritt kannst Du natürlich erst dann gehen, wenn Du alle Anpassungen durchgeführt hast.

Jetzt ist das Child-Theme grundsätzlich einsatzfähig, sodass Du es nun hochladen und in der Admin-Oberfläche im Menüpunkt **Design > Themes** aktivieren kannst.



Ein Klick genügt und Dein neues Child-Theme ist aktiv.

Erst einmal ändert sich damit gar nichts, jedenfalls nichts Sichtbares. Denn noch hast Du ja keine Änderungen im Child-Theme vorgenommen. Um zu sehen, ob Du erfolgreich warst, öffne die **style.css** und füge dort eine neue Formatierung ein, beispielsweise:

```
body {color: blue !important;
}
```

Nachdem Du die veränderte **style.css** auf dem Server gespeichert hast, sollte nun der Fließtext auf Deiner Website in Blau erscheinen.

Falls Du in WordPress mit eigenen Menüs arbeitest, gibt es nach der Aktivierung eines Child-Themes manchmal Probleme. Die behebst Du ganz einfach, wenn Du unter **Design > Menüs** das jeweilige Menü auswählst und einmal auf **Menü speichern** klickst.

Sofern vorhanden, solltest Du individuelle Menüs nach Aktivierung des Child-Themes einmal neu speichern.

## Systematik von Child-Themes

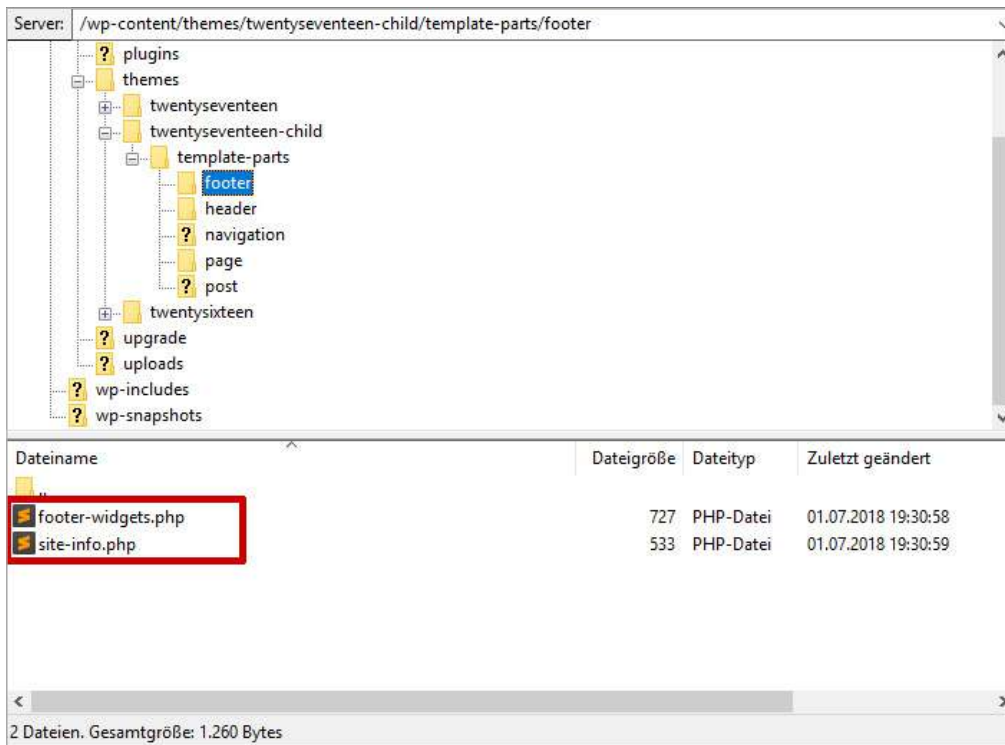
Die beiden Dateien **style.css** und **functions.php** haben in Child-Themes eine Sonderstellung: Die Einträge in der **style.css** lädt WordPress zusätzlich zu den äquivalenten Dateien des Parent-Themes. Enthält das Child-Theme also Formatierungen für dieselben Elemente wie das Parent-Theme, werden letztere durch das Child-Theme überschrieben.

Bei der **functions.php** lädt dagegen immer zuerst die Datei des Child-Themes, danach die des Parent-Themes. Funktionen mit gleichem Namen werden vom Parent-Theme also überschrieben. Du solltest daher darauf achten, dass die Funktionen in Deinem Child-Theme keine Namen benutzen, die bereits in der **functions.php** des Parent-Themes vorkommen. Ein Hinweis für erfahrene Anwender: Gleichnamige Funktionen können im Child-Theme die ursprünglichen Funktionen komplett ersetzen, indem Du sie über **remove\_action()** beziehungsweise **remove\_filter()** von dem Hook löst, dem sie zugeordnet sind, oder indem Du im Child-Theme eine höhere Priorität festlegst.

Alle anderen Dateien, die Du in Deinem Child-Theme einsetzt, ersetzen die äquivalenten Dateien des Parent-Themes komplett. Sie werden anstatt und nicht zusätzlich zu den Dateien aus dem Parent-Theme geladen. WordPress prüft also, ob eine entsprechende Datei im Child-Theme vorhanden ist und gibt selbiger gegebenenfalls den Vorzug.

Liegen die Original-Dateien in einem Unterverzeichnis, müssen die entsprechenden Dateien im Child-Theme ebenfalls in einem identischen Unterverzeichnis liegen.

Beispiel: Die Dateien **footer-widgets.php** und **site-info.php** des Themes Twenty Seventeen liegen im Unterverzeichnis **/wp-content/themes/twentyseventeen/template-parts/footer**. Um sie zu ersetzen, müssen die entsprechenden Dateien des Child-Themes einen äquivalenten Pfad haben: **/wp-content/themes/twentyseventeen-child/template-parts/footer**.



Alle Dateien mit Ausnahme von style.css und functions.php ersetzen die Dateien des Parent-Themes komplett.

## Theme-Updates prüfen

Du hast jetzt alle Möglichkeiten in der Hand, um Gestaltung und Funktionalität Deines WordPress-Themes zu verändern, ohne das zu Grunde liegende Parent-Theme anzufassen. Dieses kannst Du also bedenkenlos aktualisieren, wenn Updates dazu erscheinen.

Aber natürlich entwickelt sich jedes Theme weiter, sodass Updates auch Veränderungen betreffen können, die Du im Child-Theme gemacht hast. Dateien aus dem Parent-Theme, die Du übernommen und verändert hast, könnten Sicherheitslücken enthalten, die auch beim Update des Parent-Themes nicht behoben werden, weil WordPress weiterhin die – dann veraltete – Kopie in Deinem Child-Theme verwendet.

Deshalb ist es wichtig, dass Du bei jedem Update des Parent-Themes in dessen Dokumentation schaust und prüfst, was sich dort verändert hat, um notfalls auch Dein Child-Theme anzupassen.

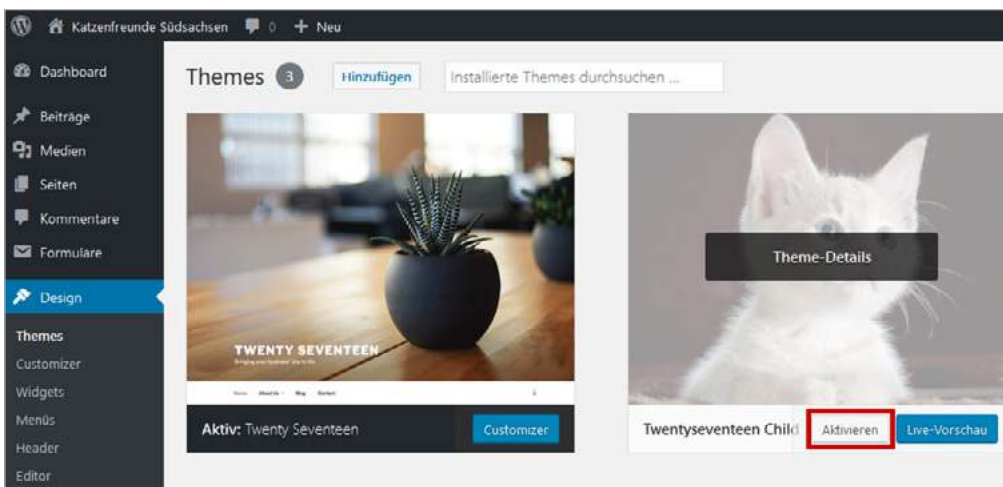
## Minimal-Prinzip

Der wichtigste Tipp für Child-Themes lautet daher: Beschränke Dich bei Veränderungen auf das absolut Nötige. Je weniger Parent-Theme-Dateien Du übernimmst und je weniger Veränderungen Du am Design vornimmst, desto unwahrscheinlicher sind spätere Konflikte – und desto unwahrscheinlicher ist es, dass Du bei Theme-Updates Anpassungen vornehmen musst.

Das geringste Konfliktpotenzial haben die CSS-Anweisungen in der **style.css**. Wenn Du also damit auskommst, ausschließlich hier Anpassungen vorzunehmen, kannst Du Dir eine umfangreichere Prüfung von Theme-Updates sparen.

## Viel Spaß beim Gestalten!

Child-Themes werden gerne unterschätzt, sind aber ein sehr mächtiges und praktisches Feature von WordPress. Du hast damit alle Freiheiten, das Theme anzupassen, ohne das Haupt-Theme zu verändern. Beachten musst Du lediglich, dass zu viele und zu umfassende Veränderungen irgendwann dennoch zu Konflikten führen werden. In solchen Fällen kann es sinnvoll sein, lieber gleich zu einem anderen Theme zu wechseln, das schon in seiner Grundstruktur besser zu Deinen Anforderungen passt. Für kleinere Design-Anpassungen sind Child-Themes dagegen sehr unkompliziert und sicher.



## 2.2 6 Schönheitstipps für das WordPress-Theme Twenty Nineteen

Das Standard-Theme von WordPress kann mehr als nur Standard. Mit einfachen CSS-Anweisungen machst Du Twenty Nineteen noch schöner.

Twenty Nineteen verfügt über ein minimalistisches, einspaltiges Layout und eignet sich damit vor allem für Blogs. Das Design kannst Du im Backend über den Customizer anpassen. Mit Cascading Style Sheets (CSS) hast Du jedoch viel mehr Möglichkeiten. Mit wenigen Code-Zeilen kannst Du zum Beispiel die Farben und Schriften individuell anpassen.

**Hinweis:** Am besten verwendest Du für die folgenden Anpassungen ein Child Theme. Alternativ kannst Du die CSS-Anweisungen auch in das WordPress-Backend eingeben (Design > Customizer > Zusätzliches CSS).

### Tipp 1: Farbfilter entfernen

Standardmäßig sind die Beitragsbilder auf der Startseite mit einem blauen Overlay versehen. Im Backend kannst Du die Farbe ändern oder den Filter unsichtbar machen (**Design > Customizer > Farben**). Das funktioniert jedoch nur bei den Bildern auf der Startseite – in den Beiträgen werden die Bilder weiterhin abgedunkelt angezeigt.



Beitragsbild mit Filter (links) und ohne

Um das zu ändern, entfernst Du im Customizer zunächst das Häkchen bei **Für hervorgehobene Bilder einen Filter unter Benutzung der Primärfarbe anwenden**:



Danach gibst Du folgende Zeilen als **Zusätzliches CSS** ein:

```
.site-header.featured-image::after {
background: none;
}
```

Das Abdunkeln der Bilder hat den Vorteil, dass die darüber liegende weiße Schrift besser zu lesen ist. Prüfe daher, ob die Schrift über Deinen Beitragsbildern auch ohne Filter noch lesbar ist.

## Tipp 2: Hintergrundfarbe ändern

Dank des hellen Hintergrunds lässt sich der Fließtext gut lesen. Falls Dir das Weiß aber zu langweilig ist, kannst Du die Hintergrundfarbe ganz leicht ändern, zum Beispiel in ein helles Grau:

```
#page {
background-color: #F5F5F5;
}
```



## Einstellungen > Lesen

**Deine Startseite zeigt**

Deine letzten Beiträge  
 Eine [statische Seite](#) (unten auswählen)

Homepage:

Beitragsseite:

**Blogseiten zeigen maximal**  Beiträge

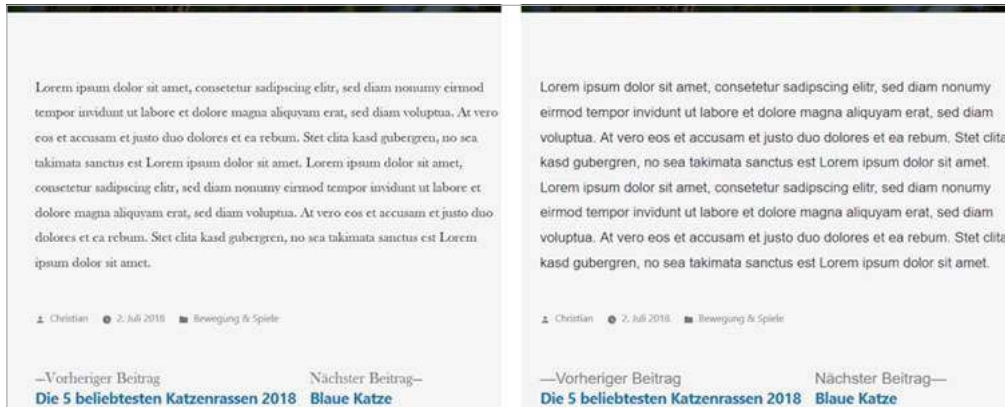
**Newsfeeds zeigen die letzten**  Einträge

Statische Seite als Startseite wählen

### Tipp 3: Schriftart anpassen

Als minimalistisches Theme setzt Twenty Nineteen auf eine ungewöhnliche Typografie. Falls Dir die Standardschriftart Hoefler Text zu exotisch ist, legst Du per CSS einfach andere Schriften fest. Mit der folgenden Anweisung sieht Dein Text aus wie beim Vorgänger-Theme Twenty Seventeen:

```
#page {
  font-family: „Libre Franklin“, „Helvetica Neue“, helvetica, arial, sans-serif;
}
```

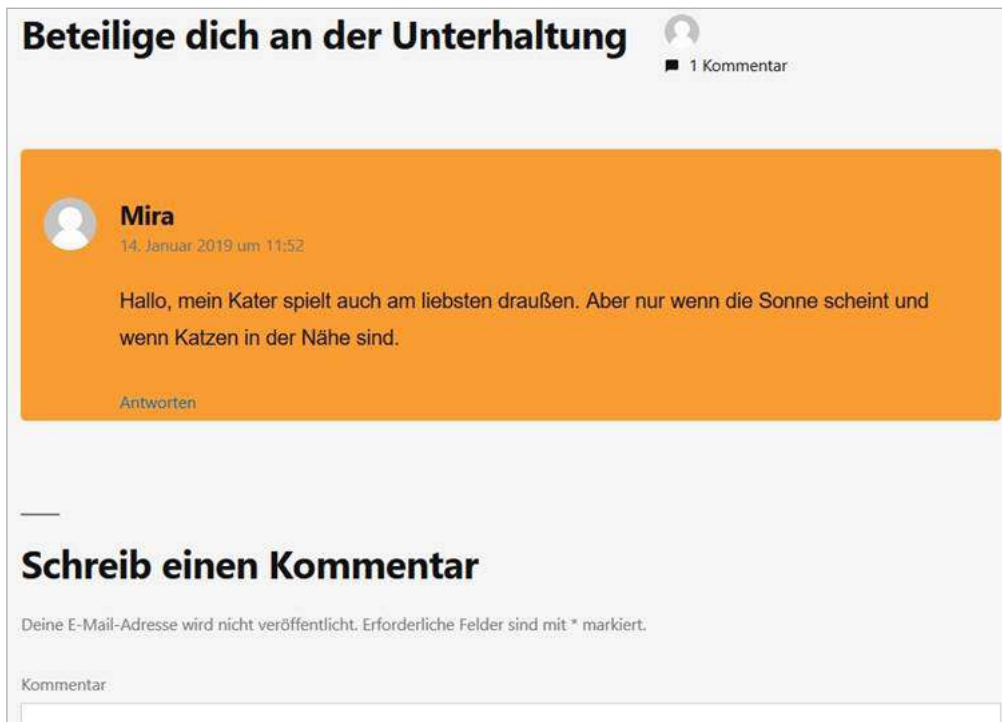


Die Serifenschrift Hoefler Text (links) wirkt klassisch, Libre Franklin (rechts) schnörkellos modern.

## Tipp 4: Hintergrundfarbe der Kommentare ändern

Im originalen Layout hängen die Kommentare ein wenig »in der Luft«, da es keine Abgrenzung vom übrigen Inhalt gibt. Abhilfe schafft hier ein farbiger Hintergrund. Über die Eigenschaft »border-radius« rundest Du zusätzlich die Ecken ab:

```
.comment-list {
  background-color: #F99C31;
  padding-left: 1rem;
  padding-top: 0.1rem;
  border-radius: 5px;
}
```



Der farbige Hintergrund hebt Kommentare hervor und lädt zum Kommentieren ein.

## Tipp 5: Hover-Effekt bei Beitragsbildern

Bei Twenty Nineteen sind die Vorschaubilder standardmäßig verlinkt. Das ist gut, weil es viele Deiner Besucher so erwarten. Noch besser wäre es, wenn die Verlinkung erkennbar wird, sobald man mit der Maus darüberfährt. Die folgenden Zeilen sorgen für einen fließenden Hover-Effekt, der die Transparenz des Bildes erhöht:

```
.entry .post-thumbnail .post-thumbnail-inner img:hover {
  transition: 0.2s ease;
  opacity: 0.9;
}
```

## Tipp 6: Widget-Schrift weniger fett

Die im Footer platzierten Widgets wirken durch die fette Schrift recht dominant. Für ein dezenteres Erscheinungsbild kannst Du den Wert von »font-weight« reduzieren:

```
.widget_archive ul li, .widget_categories ul li, .widget_meta ul li, .widget_nav_
menu ul li, .widget_pages ul li, .widget_recent_comments ul li, .widget_recent_
entries ul li, .widget_rss ul li {
  font-weight: 400;
}
```

Neueste Kommentare	Neueste Kommentare
Mira bei <a href="#">Alternativen zum Katzenkratzbaum</a>	Mira bei Alternativen zum Katzenkratzbaum
HTM bei <a href="#">Immer mehr Katzen ernähren sich vegan</a>	HTM bei Immer mehr Katzen ernähren sich vegan
Malte bei <a href="#">Immer mehr Katzen ernähren sich vegan</a>	Malte bei Immer mehr Katzen ernähren sich vegan
Marie bei <a href="#">Immer mehr Katzen ernähren sich vegan</a>	Marie bei Immer mehr Katzen ernähren sich vegan
Christian bei <a href="#">Immer mehr Katzen ernähren sich vegan</a>	Christian bei Immer mehr Katzen ernähren sich vegan

Weniger Strichstärke ist manchmal mehr (links: Original).

## Twenty Nineteen mit CSS gestalten

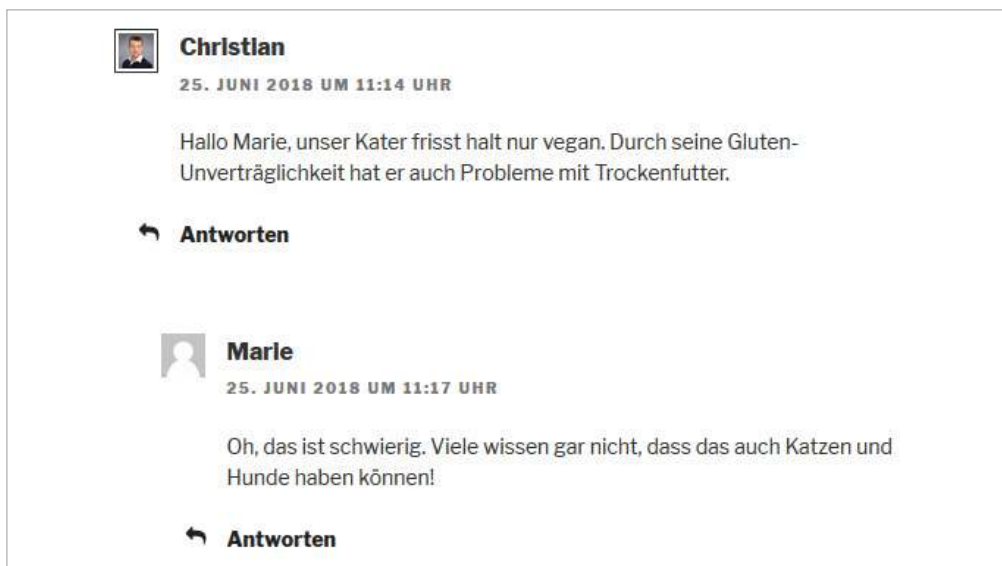
Mit CSS kannst Du Twenty Nineteen flexibel anpassen. Die WordPress-Funktion »Zusätzliches CSS« ist einsteigerfreundlich und eine bequeme Lösung für kleine Anpassungen. Im Customizer sind diese schon vor der Veröffentlichung sichtbar.

Für weitergehende Änderungen ist ein Child Theme die bessere Vorgehensweise. Hiermit kannst Du neben CSS bzw. Sass auch PHP-Code modifizieren. Die neuen WordPress-Blocks lassen sich ebenfalls über das Theme editieren. Deine Anpassungen sind dann direkt im Editor zu sehen.

Was mit CSS alles möglich ist und wie die Sprache funktioniert, erfährst Du [hier](#).

## 2.3 6 Schönheitstipps für den Kommentarbereich

Den Kommentarbereich der meisten WordPress-Themes kann man als nüchtern-funktional beschreiben. Das ist praktisch, aber wenig einladend. Doch was macht einen ansprechenden Kommentarbereich aus? Zum Beispiel eine eigene Überschrift für das Formular, hervorgehobene Autoren-Kommentare, individuelle Farben, eine Paginierung für Beiträge mit vielen Kommentaren und die Möglichkeit, Kommentare zu liken. Wenn Du das alles willst, solltest Du weiterlesen.



So sehen Kommentare bei Twenty Seventeen standardmäßig aus.

### Tipp 1: Kommentare hervorheben

Um die Kommentare und die Formularfelder vom weißen Hintergrund abzuheben, passt Du diese einfach per CSS an. Sämtliche Anweisungen kommen wie gewohnt in die **style.css** des Child-Themes.

```
/* Leser-Kommentare hervorheben */
.comment-body {
    background: #f9f9f9;
    padding: 6px;
    border-radius: 3px;
}
```

Praktischerweise vergibt WordPress für Autoren-Kommentare eine eigene Klasse namens **.bypostauthor**. Damit kannst Du diese etwa durch eine dunklere Hintergrundfarbe oder einen Rahmen von den anderen abgrenzen:

```
/* Autoren-Kommentare hervorheben */
.bypostauthor > article {
  background: #ededed;
  padding: 6px;
  border-right: #ff9900 solid 10px;
  border-radius: 3px;
}
```

Das Pfeil-Icon neben dem Antworten-Link wirkt jetzt etwas deplatziert. Mit **display:none** kannst Du das ausblenden:

```
/* Antwort-Icon ausblenden */
.comment-reply-link .icon {
  display: none;
}
```



**Christian**  
25. JUNI 2018 UM 11:14 UHR

Hallo Marie, unser Kater frisst halt nur vegan. Durch seine Gluten-Unverträglichkeit hat er auch Probleme mit Trockenfutter.

**Antworten**

**Marie**  
25. JUNI 2018 UM 11:17 UHR

Oh, das ist schwierig. Viele wissen gar nicht, dass das auch Katzen und Hunde haben können!

**Antworten**

Der Kommentar des Autors Christian wird durch einen dunkleren Hintergrund und einen Rahmen auf der rechten Seite hervorgehoben.

## Tipp 2: Überschrift für den Kommentarbereich ändern

Die Überschrift lautet standardmäßig »Schreibe einen Kommentar«. Um das zu ändern, lädst Du im Theme-Verzeichnis die Datei **comments.php** herunter und ersetzt folgende Zeile

```
comment_form();
```

durch diese:

```
comment_form(array('title_reply'=>'Sag mir Deine Meinung!'));
```

Statt »Sag mir Deine Meinung!« kannst Du natürlich auch irgendetwas anderes einfügen. Anschließend lädst Du die Datei in das Hauptverzeichnis Deines Child-Themes.

## Tipp 3: Feld für Website ausblenden

Mit den folgenden Zeilen blendest Du das Formularfeld für die Website aus. Füge den Code einfach in die **functions.php** Deines Child-Themes ein:

```
/*URL-Feld ausblenden */  
add_filter('comment_form_default_fields', 'remove_url');  
function remove_url($fields) {  
    if(isset($fields['url']))  
        unset($fields['url']);  
    return $fields;  
}
```

## Tipps 4: Formular bunter machen

Wie die Kommentare sieht auch das Kommentarformular im Auslieferungszustand etwas blass aus:

Schreibe einen Kommentar

Deine E-Mail-Adresse wird nicht veröffentlicht. Erforderliche Felder sind mit \* markiert.

**Kommentar**

  
**Name \***  
  
**E-Mail \***  
  
**Website**  
  
 Meinen Namen, E-Mail und Website in diesem Browser speichern, bis ich wieder kommentiere.

**Kommentar abschicken**



Mit wenigen CSS-Anweisungen ist aber auch das schnell erledigt:

```
/* Kommentar-Formular anpassen */
#respond.comment-respond {
  background-color: #f1f1f1;
  border-radius: 3px;
  padding: 10px;
}

.form-submit button, input[type="button"], input[type="submit"] {
  background-color: #ff9900;
}
```

So sieht das Formular aus, nachdem wir die Überschrift geändert, das Feld für die Website entfernt und Farben angepasst haben:

### Sag mir Deine Meinung!

Deine E-Mail-Adresse wird nicht veröffentlicht. Erforderliche Felder sind mit \* markiert.

**Kommentar**

  
**Name \***  
  
**E-Mail \***  
  
 Meinen Namen, E-Mail und Website in diesem Browser speichern, bis ich wieder kommentiere.

**Kommentar abschicken**

## Tipp 5: Kommentarbereich in Seiten untergliedern

Unter einer hohen Zahl an Kommentaren leiden Performance und Übersicht. In dem Fall ist eine Aufteilung des Kommentarbereichs in mehrere Seiten sinnvoll. Dazu setzt Du im Backend unter **Einstellungen > Diskussion** ein Häkchen bei »Breche Kommentare in Seiten um (...)«.

**Einstellungen > Diskussion**

**Standardeinstellungen für Beiträge**

- Versuchen, jedes in Beiträgen verlinkte Weblog zu benachrichtigen (verlangsamt das Veröffentlichen)
- Link-Benachrichtigungen von anderen Blogs (Pingbacks und Trackbacks) zu neuen Beiträgen ermöglichen
- Besuchern erlauben, neue Beiträge zu kommentieren

*(Diese Einstellungen können für jeden Beitrag individuell geändert werden.)*

**Weitere Kommentareinstellungen**

- Benutzer müssen zum Kommentieren Name und E-Mail-Adresse angeben
- Benutzer müssen zum Kommentieren registriert und angemeldet sein
- Kommentare zu Beiträgen, die älter als  Tage sind, automatisch schließen
- Verschachtelte Kommentare in  Ebenen organisieren
- Kommentare in Seiten umbrechen, mit  Top-Level-Komentaren pro Seite und die  -Seite standardmäßig anzeigen.
- Die  Kommentare sollen oben stehen

Im Frontend erscheinen nun unterhalb des Kommentarbereichs Pfeile und Seitenzahlen. Damit können Deine Besucher bequemer durch die Kommentare navigieren:



**Malte**  
25. JUNI 2018 UM 11:41 UHR

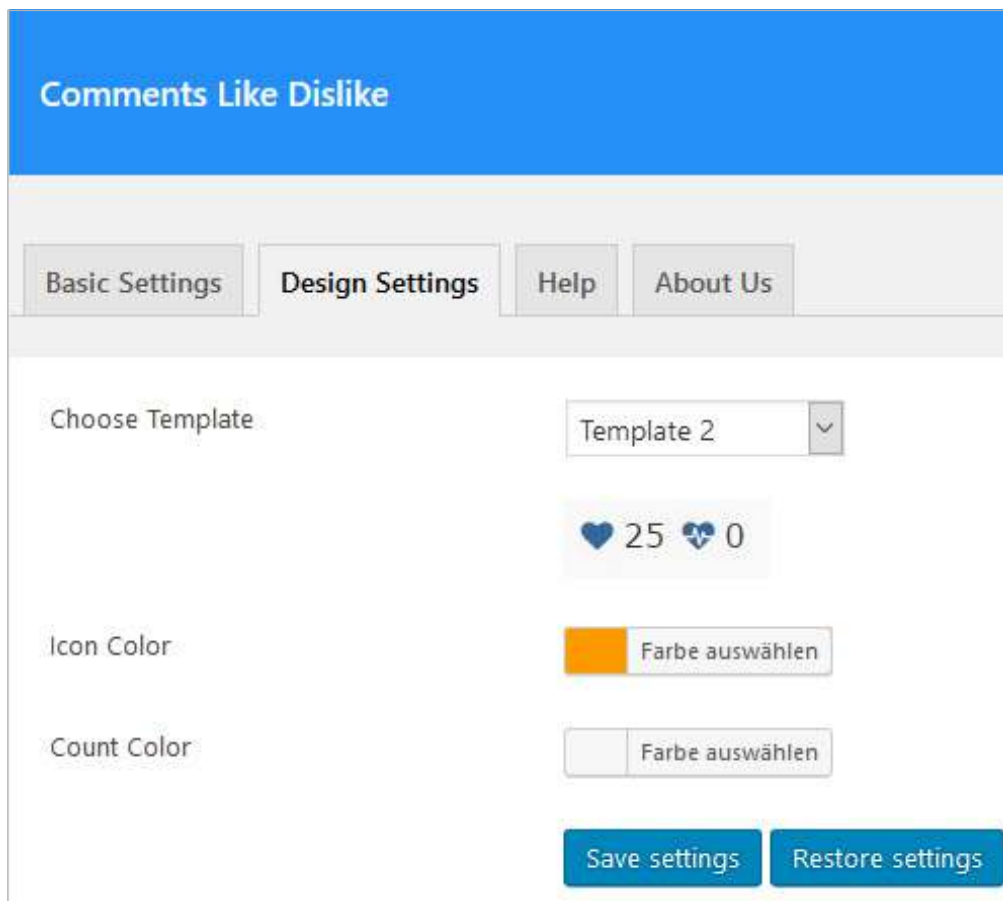
Katzengras ist auch wichtig für Katzen. Könntest Du dazu mal einen Beitrag schreiben?

**Antworten**

←
**1**
**2**

## Tipp 6: Like-Buttons hinzufügen

Liken und Favorisieren gehören zum digitalen Alltag. Dennoch: Obwohl das Bewerten anderer Posts die Interaktion steigert, verzichten die meisten Blogs auf diese Funktion. Dabei ist es mit dem [Plugin Comments Like Dislike](#) so einfach. Nachdem Du das Plugin installiert und aktiviert hast, kannst Du in den Plugin Settings unter dem Reiter **Design Settings** sogar Templates und die Farbe auswählen.




The screenshot shows the 'Comments Like Dislike' plugin settings interface. At the top, there is a blue header with the text 'Comments Like Dislike'. Below the header, there are four tabs: 'Basic Settings', 'Design Settings' (which is selected), 'Help', and 'About Us'. The 'Design Settings' section contains the following options:

- Choose Template:** A dropdown menu currently set to 'Template 2'.
- Icon Color:** A color selection area showing a blue heart icon with the number '25' and a red heart icon with the number '0'. Below this is a color picker button labeled 'Farbe auswählen' with an orange color swatch.
- Count Color:** A color picker button labeled 'Farbe auswählen' with a white color swatch.

At the bottom of the settings area, there are two buttons: 'Save settings' and 'Restore settings'.

Wir haben uns natürlich für ein orangefarbenes Herz entschieden:




**Malte**  
25. JUNI 2018 UM 11:41 UHR

Katzengras ist auch wichtig für Katzen. Könntest Du dazu mal einen Beitrag schreiben?

♥ 1

**Antworten**



**HTM**  
25. JUNI 2018 UM 11:56 UHR

Mein Kater macht gerade Diät. Er ist ein bisschen dick geworden, obwohl er sich viel bewegt. Wenn du willst, kann ich dir ein paar Low Carb Rezepte schicken!

♥ 2

**Antworten**

## 2.4 Autor-Template erstellen

Ein Autor-Template bestimmt, wie die Archiv-Seite Deiner Beiträge aussieht. Wenn Du in einem Beitrag auf Deinen Namen klickst, findest Du in der Regel lediglich eine chronologische Auflistung der letzten Beiträge. In nur wenigen Schritten verwandelst Du Dein Archiv in eine individuelle Autorensseite mit Bild, biografischen Infos und Kontaktdaten.

### 1. Template erstellen

Um die Archiv-Seite mit einem Autorenprofil zu versehen, verbindest Du den Code der `archive.php` von Twenty Seventeen mit [diesem Template](#). Mit der [Funktion `get\_avatar`](#) bindest Du das Autorenbild ein. Der komplette Code des Templates sieht dann so aus:

```
<?php
/**
 * Autor-Template
 *
 * @link https://codex.wordpress.org/Template_Hierarchy
 *
 * @package WordPress
 * @subpackage Twenty_Seventeen
 * @since 1.0
 * @version 1.0
 */

get_header(); ?>

<div class="wrap">
    <div id="primary" class="content-area">
        <main id="main" class="site-main" role="main">

<!-- This sets the $curauth variable -->

    <?php
    $curauth = (isset($_GET['author_name'])) ? get_user_by('slug', $author_name)
    : get_userdata(intval($author));
    ?>
```

```

<h2>über <?php echo $curauth->nickname; ?></h2>
  <div id="profile">
    <?php echo get_avatar( get_the_author_meta( ,ID' ), 170 ); ?>
    <?php echo $curauth->user_description; ?>
  </div>
<h3>Kontakt</h3>
<div id="contact">
  <a href="<?php echo $curauth->user_url; ?>"><?php echo $curauth->
  user_url; ?></a><br>
  <?php $user_email = get_the_author_meta(,user_email'); echo ,<a
  href="mailto:' . $user_email . ,">E-Mail schreiben</a>;?>
</div>

<h3>Beiträge von <?php echo $curauth->nickname; ?>:</h3>

<ul>

<!-- The Loop -->

<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
  <ul>
    <a href="<?php the_permalink() ?>" rel="bookmark"
    title="Permanent Link: <?php the_title(); ?>">
    <?php the_title(); ?></a>
  </ul>

  <?php endwhile; else: ?>
    <p><?php _e(,Dieser Autor hat noch keine Beiträge geschrieben. '); ?></p>

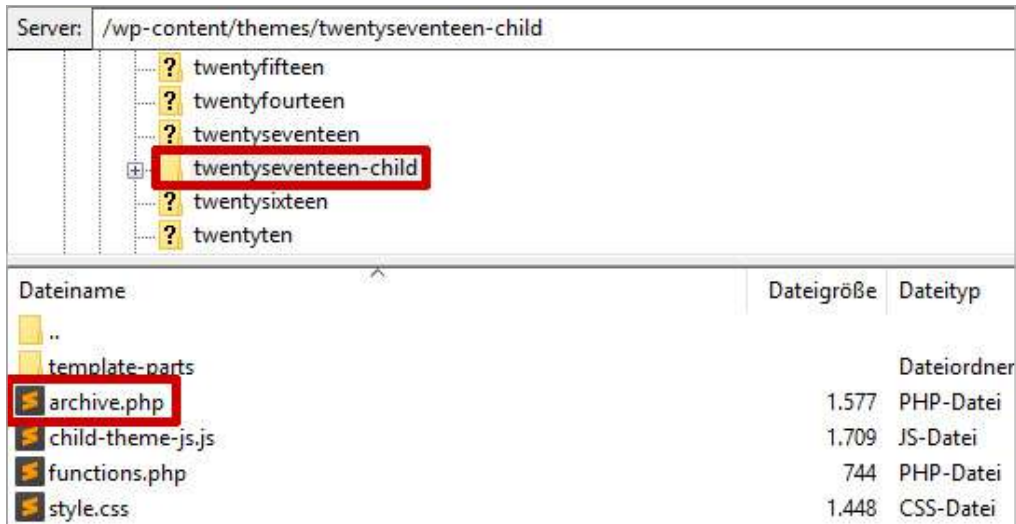
  <?php endif; ?>

<!-- End Loop -->

</ul>
</div>
<?php get_sidebar(); ?>
<?php get_footer(); ?>

```

Anschließend lädst Du die **archive.php** in das Hauptverzeichnis des Child-Themes:




Server: /wp-content/themes/twentyseventeen-child

Dateiname	Dateigröße	Dateityp
..		
template-parts		Dateiordner
<b>archive.php</b>	1.577	PHP-Datei
child-theme-js.js	1.709	JS-Datei
functions.php	744	PHP-Datei
style.css	1.448	CSS-Datei

## 2. Profil im Backend ausfüllen

Das Template kann Profilbild, biografische Angaben und Kontaktinfos nur anzeigen, wenn Du diese im Backend eingetragen hast (**Benutzer > Dein Profil**). Als Profilbild verwendet WordPress das bei [Gravatar](#) hochgeladene Foto. Damit das funktioniert, musst Du Dich dort mit der gleichen E-Mail-Adresse wie im Blog anmelden und ein Bild einstellen. Deine E-Mail-Adresse wird automatisch verlinkt und damit zur Kontaktadresse.

Spitzname <i>(erforderlich)</i>	<input type="text" value="Thomas"/>
Öffentlicher Name	<input type="text" value="Thomas"/> ▼
<b>Kontaktinfo</b>	
E-Mail <i>(erforderlich)</i>	<input type="text" value="thomas@katzenfreunde.de"/>
Website	<input type="text" value="http://katzenfreunde-suedsachsen.de"/>
<b>Über den Benutzer</b>	
Biografische Angaben	<div style="border: 1px solid #ccc; padding: 5px;">Ich bin Thomas und liebe Katzen. Am liebsten bin ich auf Reisen, mache Yoga und Sport. Freunde und Familie und Katzen sind für mich das Wichtigste im Leben. Für ein gutes Buch oder einen spannenden Film bin ich auch immer zu haben.</div>
<small>Teile ein paar biografische Informationen, um dein Profil zu ergänzen. Die Informationen könnten öffentlich sichtbar sein.</small>	
Profilbild	

Für ein vollständiges Autorenprofil brauchst Du nur die Felder im Backend auszufüllen.

### 3. CSS anpassen

Per CSS kannst Du anschließend Farben und Abstände anpassen. Die Anweisungen schreibst Du in die **style.css** Deines Child-Themes oder in den Customizer (**Design > Themes > Customizer > Zusätzliches CSS**). In unserem Beispiel färben wir die Links orange, fügen hellgraue Hintergründe und auf der linken Seite blaue Balken ein.

```
a {
  color: #ff9900;
}

#profile {
  background-color: #f4f4f4;
  padding: 8px;
  float: right;
  border-left: solid 8px #0164a5;
}

#profile img {
  float: left;
  margin-right: 12px;
}
```





```

}

#contact {
background-color: #f4f4f4;
padding: 8px;
line-height: 2;
border-left: solid 8px #0164a5;
}

```

Fertig! Links siehst Du die originale Archiv-Seite, rechts die angepasste Variante mit Profil:

<p><b>AUTOR: THOMAS</b></p> <p>Ich bin Thomas und liebe Katzen. Am liebsten bin ich auf Reisen, mache Yoga und Sport. Freunde und Familie und Katzen sind für mich das Wichtigste im Leben. Für ein gutes Buch oder einen spannenden Film bin ich auch immer zu haben.</p> <p>23. APRIL 2018</p> <p>Immer mehr Katzen ernähren sich vegan</p>  <p>&gt;Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.</p> <p><a href="#">weiterlesen</a></p>	<p>über Thomas</p>  <p>Ich bin Thomas und liebe Katzen. Am liebsten bin ich auf Reisen, mache Yoga und Sport. Freunde und Familie und Katzen sind für mich das Wichtigste im Leben. Für ein gutes Buch oder einen spannenden Film bin ich auch immer zu haben.</p> <p><b>Kontakt</b></p> <p><a href="http://katzenfreunde-suedsachsen.de">http://katzenfreunde-suedsachsen.de</a></p> <p><a href="#">E-Mail schreiben</a></p> <p><b>Beiträge von Thomas:</b></p> <ul style="list-style-type: none"> <li><a href="#">Immer mehr Katzen ernähren sich vegan</a></li> <li><a href="#">Studie: Katzen machen glücklich</a></li> <li><a href="#">Cindy läuft weg und kommt nach drei Monaten wieder</a></li> <li><a href="#">Wahl der Miss Katze 2018</a></li> <li><a href="#">Kater mag keine Mäuse fangen</a></li> <li><a href="#">Katzen brauchen Abwechslung</a></li> </ul>
--	---

## Persönlich statt Standard

Vor allem der persönliche Charakter unterscheidet einen Blog von einem Nachrichtenportal. Wenn ein Leser auf Deinen Namen klickt, will er mehr über Dich erfahren. Mit dem vorgestellten Template erfüllst Du diese Erwartung besser als mit einem Standardarchiv. Einmal eingerichtet, wirst Du das Autor-Template nicht mehr missen wollen.

## 2.5 Benutzerdefinierte Felder anlegen

Vor allem spezialisierte Blogs haben bestimmte Datentypen. In einem Kochblog beispielsweise sind das Zutaten, Zubereitungszeit und Schwierigkeitsstufe. Mit den benutzerdefinierten Feldern musst Du diese Angaben nicht mehr im Fließtext machen. Stattdessen legst Du individuelle Felder an, die ober- oder unterhalb des Beitrags angezeigt werden. Damit sparst Du Zeit und es sieht professioneller aus, wenn Du die Daten optisch vom Fließtext abgrenzt. Damit Suchmaschinen diese Inhalte besser verarbeiten, kannst Du Deine Daten auf Basis von [›schema.org](https://schema.org) strukturieren oder ein [›Plugin](#) verwenden.

Wie Du die Funktion der benutzerdefinierten Felder nutzt, zeigen wir Dir in vier Schritten:

### 1. Ansicht anpassen

Benutzerdefinierte Felder werden im Backend auf der Beitragsebene unterhalb des Editors angelegt (**Beiträge > Alle Beiträge > Bearbeiten**). Falls Du diese Option nicht findest, musst Du über die Lasche **Ansicht anpassen** das entsprechende Häkchen setzen.

**Boxen**

Revisionen  Auszug  Trackbacks senden  Benutzerdefinierte Felder  Diskussion

Kommentare  Titelform  Autor  Beitragsformat  Kategorien  Schlagwörter

Beitragsbild

**Zusätzliche Einstellungen**

Anzeige des Editors in voller Höhe zulassen und ablenkungsfreies Schreiben aktivieren.

So, take her wrap, fellas

Beitrag bearbeiten

Über die Lasche rechts oberhalb des Editors blendest Du die benutzerdefinierten Felder ein.

### 2. Benutzerdefinierte Felder anlegen

Um beim Beispiel des Kochblogs zu bleiben: Während Du die Zubereitung im Fließtext beschreibst, legst Du nun unterhalb des Editors für Zutaten, Zubereitungszeit und Schwierigkeitsgrad einfach entsprechende benutzerdefinierte Felder an.

Als **Name** trägst Du den Datentyp (z. B. »Schwierigkeitsgrad«) und daneben den für dieses Rezept gültigen **Wert** ein (z. B. »mittel«).

Name	Wert
Schwierigkeitsgrad	mittel
Zubereitungszeit	30 Minuten
Zutaten	300 g Margarine, 300 g Butter, 300 g Mehl, 5 Eier, 1 x Backpulver, 1 x Vanillezucker, 3 x Puddingpulver, 800 g Schmand, 8 EL Zucker, 2 x Tortenquass, 2 kg Erdbeeren

Übrigens: Die Namen werden dabei gespeichert und stehen per Dropdown-Menü auch für andere Beiträge zur Verfügung. Das spart Zeit und stellt sicher, dass die Datentypen in den Beiträgen einheitlich benannt werden.

### 3. Felder im Beitrag anzeigen

Die benutzerdefinierten Felder zeigt WordPress erst mit der Funktion `<?php the_meta(); ?>` an. Dazu fügst Du den Code innerhalb des **Loops** ein. Der Loop bzw. die »Schleife« ist ein PHP-Code, mit dem WordPress die Inhalte anzeigt.

Wo die Funktion genau hingehört, hängt vom verwendeten Theme ab. Themes sind in der Regel kommentiert (z. B. »/\* Start the Loop \*/«), sodass Du nicht lange suchen musst. Bei Twenty Seventeen zum Beispiel bzw. dessen Child-Theme kannst Du die Funktion in die Datei **single.php** oberhalb des Kommentars »// If comments are open or we have at least one comment, load up the comment template.« einfügen:

```

get_header(); ?>
<div class="wrap">
  <div id="primary" class="content-area">
    <main id="main" class="site-main" role="main">

      <?php
      /* Start the Loop */
      while ( have_posts() ) : the_post();

          get_template_part( 'template-parts/post/content', get_post_format() );

          // Custom Fields
          the_meta();

          // If comments are open or we have at least one comment, load up the comment template.
          if ( comments_open() || get_comments_number() ) :
              comments_template();
          endif;

          the_post_navigation( array(
              'prev_text' => '<span class="screen-reader-text">' . __( 'Previous Post', 'twentyseventeen' ) . '</span> <span class="nav-title"><span class="nav-title-icon">
              </span>%title</span>',
              'next_text' => '<span class="screen-reader-text">' . __( 'Next Post', 'twentyseventeen' ) . '</span> <span class="nav-title">%title<span class="nav-title-icon-wrappe
          ) );

          endwhile; // End of the loop.
      ?>

    </main><!-- #main -->
  </div><!-- #primary -->
  <?php get_sidebar(); ?>
</div><!-- .wrap -->

<?php get_footer();

```

Sobald Du den Beitrag neu lädst, findest Du die benutzerdefinierten Felder zwischen Beitrag und Kommentarbereich:



Jetzt nur noch abkühlen lassen und fertig ist der Kuchen. Da kann natürlich noch Sahne drauf. Lass es Dir schmecken! 😊

#### # KUCHEN

- Schwierigkeitsgrad: mittel
- Zubereitungszeit: 30 Minuten
- Zutaten: 300 g Margarine, 300 g Butter, 300 g Mehl, 5 Eier, 1 x Backpulver, 1 x Vanillezucker, 3 x Puddingpulver, 800 g Schmand, 8 EL Zucker, 2 x Tortenguss, 2 kg Erdbeeren

## 4. Daten vom Fließtext abgrenzen

Sofern im Theme nichts Anderes definiert wurde, sehen die benutzerdefinierten Felder genauso aus wie der Rest. Um das zu ändern, kannst Du der **style.css** Deines Child-Themes zum Beispiel folgende Zeilen hinzufügen:

```
/* Custom Fields */
.post-meta {
  margin-top: 10px;
  background-color: #ff9900;
  color: #ffffff;
  list-style: none;
  padding: 5px;
}
.post-meta-key {
  font-weight: bold;
  font-variant: small-caps;
  list-style: none;
}
```

Fertig:



Jetzt nur noch abkühlen lassen und fertig ist der Kuchen. Da kann natürlich noch Sahne drauf. Lass es Dir schmecken! 😊

# KUCHEN

**SCHWIERIGKEITSGRAD:** mittel

**ZUBEREITUNGSZEIT:** 30 Minuten

**ZUTATEN:** 300 g Margarine, 300 g Butter, 300 g Mehl, 5 Eier, 1 x Backpulver, 1 x Vanillezucker, 3 x Puddingpulver, 800 g Schmand, 8 EL Zucker, 2 x Tortenguss, 2 kg Erdbeeren

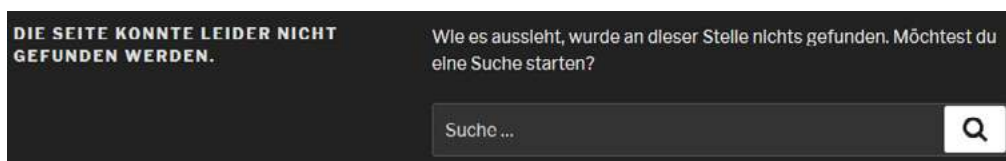
## Mehr Struktur mit benutzerdefinierten Feldern

Es macht wenig Spaß, jedes Mal dieselben Datentypen aufzulisten. Mit der vorgestellten Lösung kannst Du Zutaten, Literaturangaben und Produkteigenschaften bequem in Deine Beiträge integrieren und ansprechend darstellen. Praktisch ist der damit einhergehende Wiedererkennungseffekt: Leser, die sich für die Details interessieren, finden sie auf einen Blick – einheitlich und in jedem Beitrag an der gleichen Stelle.

## 2.6 404-Fehlerseite individuell gestalten

Ob gelöschte Datei oder fehlerhafte URL: Der 404-Error hat verschiedene Ursachen. Wie Du das Problem löst, erfährst Du in [Kapitel 5.3](#). In manchen Fällen lässt sich der Fehler jedoch nicht vermeiden – etwa bei einem fehlerhaften Link oder wenn die URL von einem Nutzer falsch eingegeben wurde. WordPress gibt dann standardmäßig folgende Meldung aus: »Die Seite konnte leider nicht gefunden werden.«

Diese Meldung ist leider nicht besonders ansprechend. Allerdings wirst Du derartige 404-Templates in den meisten Themes vorfinden. Bei einer professionellen Fehlerseite mit individuellem Text, Bild und Widgets bleiben Deine Besucher eher auf Deiner Website.



Standard-404-Fehlerseite beim Theme Twenty Seventeen

### Fehlermeldung anpassen

Als Erstes solltest Du den Wortlaut der Fehlermeldung an den Inhalt Deiner Website anpassen.

Die entsprechenden Sprachdateien (.po- und .mo-Dateien für mehrsprachige Texte) werden bei WordPress nicht im Theme-Ordner, sondern in einem zentralen Language-Verzeichnis gespeichert. Würdest Du die Sprachdatei einfach ändern, gingen die Änderungen bei einem Update verloren. Um das Problem zu umgehen, legst Du eine neue Sprachdatei an.

Am einfachsten geht das mit dem Plugin [Loco Translate](#). Navigiere nach der Installation im WordPress-Backend über **Loco Translate > Themes** zu Deinem Theme (**laut Plugin-Vorgabe nicht Child-Theme!**). Klicke oben auf **New Language**, wähle als Sprache **Deutsch** und als Ort **languages/loco/themes/twentyseventeen-de\_DE.po** (bzw. die entsprechende .po-Datei Deines Parent-Themes). Damit erstellt das Plugin eine zusätzliche Sprachdatei und legt diese in einem separaten Unterverzeichnis ab.

Themes / Twenty Seventeen / New language


Overview Setup Advanced Debug

### Initializing new translations in "twentyseventeen"

72 strings found in twentyseventeen.pot

1. Choose a language:

WordPress language:  Custom language:

 Deutsch

2. Choose a location: [? What's this?](#)

*Custom:*  languages/loco/themes/twentyseventeen-de\_DE.po

*System:*  languages/themes/twentyseventeen-de\_DE.po

*Author:*  themes/twentyseventeen/de\_DE.po

[Start translating](#)

Wichtig: Die Übersetzung musst Du im Loco-Ordner speichern.

Um die 404-Meldung zu modifizieren, navigierst Du nun über **Loco Translate > Themes** zu Deinem Theme und klickst unterhalb des Sprach-Icons auf **Edit**. Wähle die neue Sprachdatei im Custom-Folder:

Language	Translation progress	Pending	File info	Last modified	Folder
 Deutsch	<div style="width: 3%;"></div> 3%	70	twentyseventeen-...	27. September 20...	Custom
<a href="#">edit</a> <a href="#">view</a> <a href="#">copy</a> <a href="#">info</a> <a href="#">delete</a>					
 Deutsch	<div style="width: 100%;"></div> 100%	0	twentyseventeen-...	27. September 20...	System

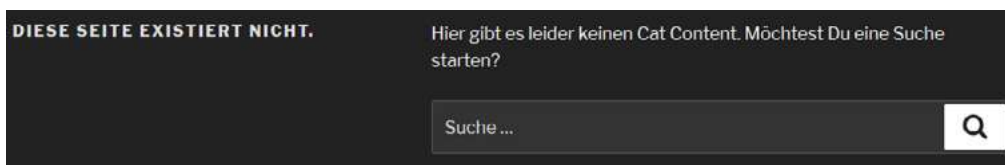
Anschließend gibst Du den Wortlaut der 404-Fehlermeldung Deines Themes oben in den Filter ein. Das Plugin zeigt daraufhin die entsprechende Textstelle an, die Du dann per Klick auswählen und in den Feldern unten bearbeiten kannst.





Filterfunktion in Loco Translate

Fertig. Auf diese Weise kannst Du die beiden Standardtexte und alle anderen Sprachdateien von Themes und Plugins beliebig anpassen:



## 404-Template erstellen

Um Bilder oder Widgets hinzuzufügen, benötigst Du ein 404-Template. Die Datei **404.php** liegt im Hauptverzeichnis Deines Themes. Lege eine Kopie der Datei in das Hauptverzeichnis Deines Child-Themes. Falls Dein Theme über keine **404.php** verfügt, erstellst Du diese Datei selbst. Dazu kannst Du folgenden Code verwenden (ersetze »twentyseventeen« mit dem Namen Deines Themes):

```
<?php
get_header(); ?>
<div class="wrap">
  <div id="primary" class="content-area">
    <main id="main" class="site-main" role="main">
      <section class="error-404 not-found">
        <header class="page-header">
          <h1 class="page-title"><?php _e( 'Oops! That page can't be
            found.', 'twentyseventeen' ); ?></h1>
        </header><!-- .page-header -->
        <div class="page-content">
          <p><?php _e( 'It looks like nothing was found at this location. Maybe try
            a search?', 'twentyseventeen' ); ?></p>
          <?php get_search_form(); ?>
        </div><!-- .page-content -->
      </section>
    </main>
  </div>
</div>
```

```

</section><!-- .error-404 -->
</main><!-- #main -->
</div><!-- #primary -->
</div><!-- .wrap -->
<?php get_footer();

```

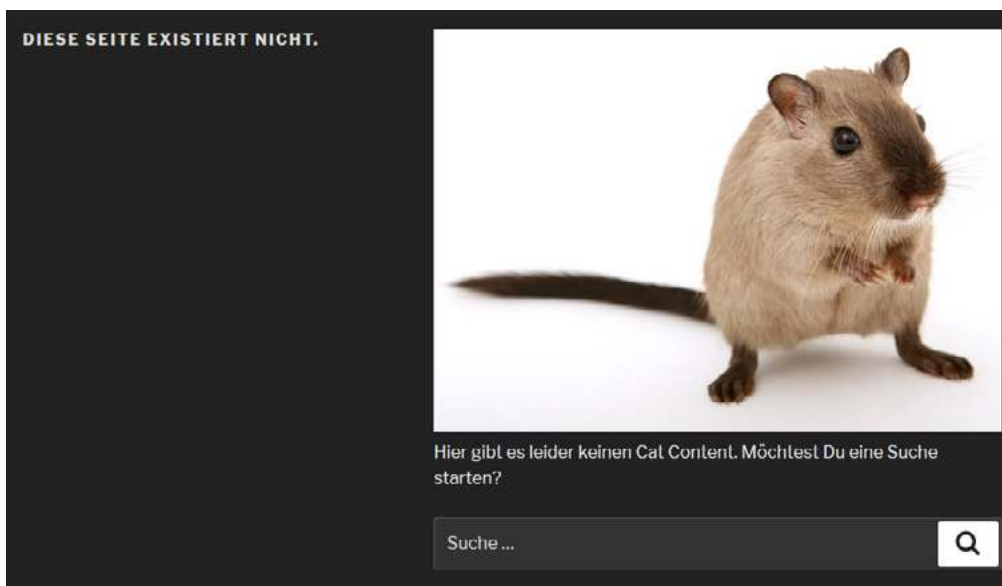
## Bild einfügen

Ist das 404-Template erstellt, kannst Du Bilder und Widgets in die Fehlerseite integrieren. Mit einem Bild wirkt Deine 404-Seite gleich viel ansprechender. Lade dazu die **404.php** im Hauptverzeichnis Deines Parent-Themes herunter und füge einfach einen Image-Tag mit der URL Deines Bildes in der Mediathek ein. Zum Beispiel:

```



```



Mouse Content

## Widgets hinzufügen

Um Deine Besucher bei der Stange zu halten, kannst Du zusätzlich Widgets hinzufügen. Hier eignet sich am ehesten die Anzeige Deiner aktuellen Beiträge. So können die Besucher direkt zu den Inhalten springen.

Gib den nachfolgenden Code in die **functions.php** Deines Child-Themes ein, um einen Widget-Bereich für die 404-Fehlerseite zu registrieren (in diesem Beispiel für ein Twenty-Seventeen-Child-Theme):


```
function twentyseventeen_child_widgets_init() {
    register_sidebar(array(
        ,name'           => ,404-Fehlerseite',
        ,id'             => ,404-fehlerseite',
        ,description'    => ,Widgets hier hinzufügen, um sie auf der 404-Fehlerseite
        anzuzeigen.',
        ,before_widget'  => ,<div class="404-widget">',
        ,after_widget'   => ,</div>',
        ,before_title'   => ,<h2>',
        ,after_title'    => ,</h2>',
    ));
}
add_action( ,widgets_init' , ,twentyseventeen_child_widgets_init' );
```

Kopiere anschließend die folgenden Zeilen in die **404.php** des Child-Themes, um WordPress mitzuteilen, wo der Widget-Bereich angezeigt werden soll:

```
<?php if ( is_active_sidebar( ,404-fehlerseite' ) ) :?>
<?php dynamic_sidebar( ,404-fehlerseite' );?>
<?php endif;?>
```

Nun erscheint im Backend unter **Design > Widgets** der neue Bereich »404-Fehlerseite«. Ziehe dort wie gewohnt per Drag & Drop ein oder mehrere Widgets hinein. Mit den aktuellen Beiträgen sieht das anschließend so aus:

**DIESE SEITE EXISTIERT NICHT.**



Hier gibt es leider keinen Cat Content. Möchtest Du eine Suche starten?

Suche ...

Hier gibt es aktuellen Cat Content:

- Alternativen zum Katzenkratzbaum
- Die 5 beliebtesten Katzenrassen 2018
- Immer mehr Katzen ernähren sich vegan
- Studie: Katzen machen glücklich
- Cindy läuft weg und kommt nach drei Monaten wieder

## 404-Fehlerseiten können lustig sein

Wie Deine Besucher auf eine nicht gefundene Seite reagieren, hängt von Deiner Kreativität ab. Weil der 404-Fehler nicht dramatisch ist, dürfen 404-Meldungen **› auch lustig sein**. Solange der Error eine Ausnahme bleibt und Du Deinen Besuchern Optionen anbietest, bleiben Sie Dir höchstwahrscheinlich erhalten.

Über die gezeigten Schritte hinaus kannst Du Dein Template natürlich weiter anpassen – **› vom Entfernen der Listenpunkte bis hin zum Fullwidth-Layout**. Vielleicht sehen Deine Besucher den Error dann ebenfalls mit Humor. Und wenn nicht, leitest Du sie einfach **› per Plugin** auf die Startseite weiter.

## 2.7 Landing Page mit einem Seiten-Template erstellen

Auf einer Landing Page sollen Besucher ohne Umwege und Ablenkung ein bestimmtes Produkt kaufen oder sich zu einer Veranstaltung anmelden können. Um dieses Ziel zu erreichen, muss eine Landing Page zielgruppenorientiert, argumentativ und auf das Wesentliche reduziert sein. Mit dem Klick auf den sogenannten Call-to-Action-Button wird aus dem Besucher schließlich ein Kunde.

WordPress hat mit den Seiten-Templates die passende Funktion bereits an Bord. In dieser Anleitung erfährst Du in vier Schritten, wie Du ein Seiten-Template für das Theme Twenty Seventeen bzw. dessen Child-Theme anlegst. Dazu bauen wir als Beispiel eine einfache Landing Page, über die sich Besucher in unserem Beispiel für eine Yoga-Veranstaltung anmelden können.

### Schritt 1: Seiten-Template erstellen

Um das Template zu erstellen, lade die Datei **page.php** aus dem Verzeichnis Deines Parent-Themes (`/wp-content/themes/twentyseventeen`) herunter und benenne sie in **landing-page.php** um. Um daraus ein Seiten-Template zu machen, ersetzt Du den Kommentar am Anfang durch folgenden:

```
/**
 * Template Name: Landing Page
 *
 * @package Twenty Seventeen Child
 */
```

Anschließend lädst Du die Datei in das Hauptverzeichnis Deines Child-Themes.

Für die Landing Page erstellst Du nun eine Seite und präsentierst dort Dein Angebot. In der Seiten-Ansicht im Backend befindet sich rechts unterhalb von **Veröffentlichen** der Bereich **Seiten-Attribute**. Dort wechselst Du im Dropdown-Menü vom **Standard-template** auf **Landing Page**.

### Seiten-Attribute ▲

**Eltern**

Hauptseite (keine Übergeordnete) ▼

**Template**

Standardtemplate ▼

Standardtemplate

Landing Page

0

Brauchst du Hilfe? Benutze den Tab „Hilfe“ oben am Bildschirm.

## Schritt 2: Kontaktformular einfügen

Damit sich die Besucher für die Veranstaltung anmelden können, benötigst Du ein Kontaktformular. Empfehlenswert ist das **Plugin Contact Form 7** an, da es laufend gepflegt wird und sich flexibel anpassen lässt. Nach der Installation kannst Du im Backend unter **Formulare** ein neues hinzufügen. Über die Buttons wählst Du einfach die gewünschten Felder: Name, E-Mail-Adresse, Telefon. Den »Senden«-Button solltest Du so umbenennen, dass er zu Deinem Angebot passt (z. B. »Anmelden«, »Kaufen«, »Download«).

**Yoga Mallorca 2018**

Kopiere diesen Shortcode und füge ihn in den Inhalt deines Beitrag, Seite oder Text-Widget ein:

`[contact-form-7 id="338" title="Yoga Mallorca 2018"]`

**Formular** | E-Mail | Meldungen | Zusätzliche Einstellungen

**Formular**

Text | E-Mail | URL | Tel. | Zahl | Datum | Textfeld | Dropdown-Menü | Kontrollkästchen | Radio-Buttons | Zustimmung | Quiz | reCAPTCHA | Datei | Senden

```

<label> Name (Pflichtfeld)
[ text* your-name ] </label>

<label> E-Mail-Adresse (Pflichtfeld)
[ email* your-email ] </label>

<label> Telefon
[ tel tel-921 ] </label>

[ submit "Anmelden" ]

```

**Speichern**

Um den Button des Formulars umzubenennen, änderst Du das Tag, z. B. [submit »Anmelden«].

Nach dem Speichern kopierst Du den blau markierten Code oberhalb des Editors in die Landing Page. Diese sieht in unserem Fall anschließend so aus:

Ohne CSS-Anpassungen: Eine typische WordPress-Seite mit Header und Navigation.

### Schritt 3: CSS anpassen

Mit dem nachfolgenden CSS-Code sieht das Ganze gleich viel besser aus. Hier sind die Anpassungen im Überblick:

- komplette Breite des Content-Bereichs wird genutzt
- Header und Navigation sind ausgeblendet
- Hintergrundbild vorhanden
- Content-Bereich mit weißer Farbe und Transparenz
- Überschrift, Button und Link zum Impressum angepasst

WordPress vergibt für Seiten-Templates neue Klassen nach dem Schema **body.page-template-TEMPLATENAME**. Damit kannst Du die Bereiche flexibel anpassen:

```
/* Anpassungen Landing Page */

/* Breite des Content-Bereichs nutzen */
body.page-template-landing-page #content #primary {
  max-width: 100%;
}

body.page-template-landing-page #content #primary .entry-header,
body.page-template-landing-page #content #primary .entry-content {
  float: none;
  width: 100%;
}

/* Header ausblenden */
body.page-template-landing-page .site-header {
  display: none;
}

/* Hintergrundbild einfügen */
body.page-template-landing-page {
  background-image: url(„/wp-content/themes/twentyseventeen-child/bg_image.
  jpg“);
}

body.page-template-landing-page .site-content-contain {
  background: none !important;
}

/* Content-Bereich mit weißer Farbe hinterlegen */
body.page-template-landing-page #content .wrap {
  display: flex;
  align-items: center;
  justify-content: center;
  background-color: rgba(255,255,255,0.5);
  border-radius: 3px;
}

/* Überschrift anpassen */
body.page-template-landing-page .entry-title {
  font-size: 2em !important;
  margin-top: 0.6em;
}
```



```

    color: #ff9900 !important;
}

/* Button anpassen */
body.page-template-landing-page .wpcf7-submit {
    width: 100%;
    background: #00AA72 !important;
    color: #fff !important;
    padding: 20px;
}

/* Linkfarbe Impressum weiß */
body.page-template-landing-page a {
    color: #ffffff;
}

```

## Schritt 4: Impressum einfügen

Auf einer Landing Page muss – wie auch auf allen anderen Seiten einer Website – ein Link zum Impressum vorhanden sein. Mit diesem Link kannst Du gleich den Hinweis »Stolz präsentiert von WordPress« im Footer ersetzen. Das sorgt für weniger Ablenkung und Ausstiegsmöglichkeiten. Bei Twenty Seventeen liegt die entsprechende Datei **site-info.php** im Ordner **template-parts** (Unterverzeichnis footer). Bei Twenty Nineteen findet sich der Eintrag in der Datei **footer.php**. Am besten lädst Du einfach das komplette Verzeichnis **template-parts** samt Inhalt herunter und ersetzt den Code in der **site-info.php** bzw. **footer.php** durch diesen:

```

<?php
/**
 * Displays footer site info
 *
 * @package WordPress
 * @subpackage Twenty_Seventeen
 * @since 1.0
 * @version 1.0
 */

?>
<div class="site-info">
    <a href="http://DEINE-URL/">Impressum</a>
</div><!-- .site-info -->

```

Die Verweise zur Datenschutz-Seite, den AGB oder der Widerrufsbelehrung fügst Du genauso wie das Impressum innerhalb der Klasse »site-info« ein. Die Links erscheinen dann nebeneinander im Footer. Danach lädst Du den Ordner **template-parts** in das Hauptverzeichnis Deines Child-Themes.

Hier ist das Ergebnis unserer Yoga-Landing-Page:



Und so sieht das auf dem Smartphone aus:



The image shows a smartphone screen displaying a registration form. At the top, the status bar shows signal strength, Wi-Fi, 79% battery, and the time 20:10. The form title is "Frühstück, Unterricht". Below the title, the location is "Ort: Cala Llombards, Tofu Hotel" and the dates are "Zeit: 23.-25. März 2019". A phone number "Fragen? 0123-123456789" is listed. The form contains three input fields: "Name (Pflichtfeld)", "E-Mail-Adresse (Pflichtfeld)", and "Telefon". A green button labeled "Anmelden" is positioned below the input fields. At the bottom, a note states "\* Die Teilnehmerzahl ist auf 20 Personen begrenzt!". The Android navigation bar is visible at the very bottom.

## Landing Page in vier Schritten

Mit ein bisschen Geduld ist es gar nicht so schwer, eine einfache Landing Page zu entwickeln. Das Praktische dabei: Hast Du das Seiten-Template einmal angelegt, kannst Du das Design unabhängig von den anderen Seiten beliebig anpassen. Das bedeutet, dass sich Funktionen und Layout komplett vom Rest der Website unterscheiden können. Die Funktion der Seiten-Templates von WordPress ist deshalb wie geschaffen für individuelle Landing Pages.

## 2.8 Eigenes Theme entwickeln: Bootstrap-Navigation in WordPress implementieren

Bootstrap ist ein Open-Source-Framework für Webdesigner mit fertigen Buttons, Navigationsleisten, Grid-Systemen, Tabellen und vielem mehr. Als Basis dienen bereits formatierte CSS-Komponenten, typografische Elemente und JavaScript. Die benötigten Dateien und Code-Blöcke werden auf [GitHub](#) veröffentlicht und können frei verwendet werden. Wir zeigen Dir am Beispiel einer Navigationsleiste, wie Du Komponenten in WordPress einfügst.

### 1. Basis-Theme erstellen

Zum Testen benötigst Du als Erstes ein möglichst einfaches Theme ohne Schnickschnack. Für solche Zwecke gibt es etwa das [Starter-Theme underscores von Automattic](#): Einfach einen Namen eingeben (in unserem Beispiel »meintheme«) und generieren lassen. Das als .zip-Datei gepackte Theme kannst Du dann über das Dashboard hochladen und aktivieren.

Aus Sicherheitsgründen sollte mindestens ein alternatives Theme aktiviert sein, auf welches WordPress im Notfall zurückgreifen kann.

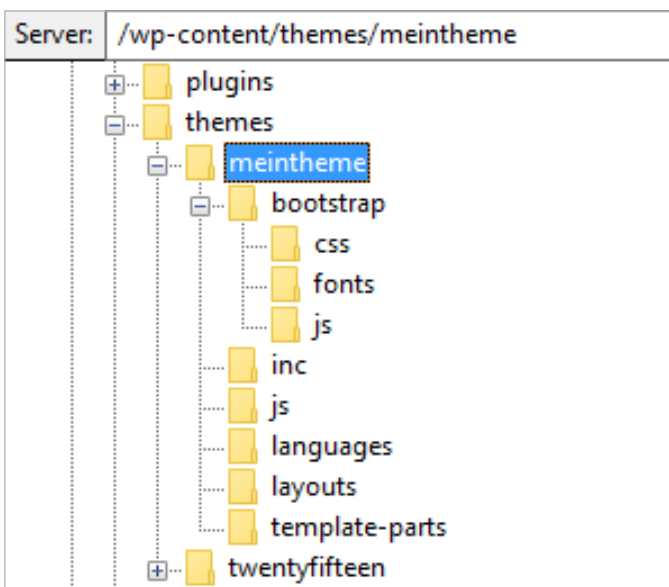


The image shows a web interface for generating a theme. It features a light green background with a dark blue header. In the center, there is a white text input field containing the text 'meintheme'. To the right of the input field is a dark blue button with the text 'GENERATE' in white. Below the input field, the text 'Advanced Options' is visible. Below the button, the text 'Based on \_s from github' is visible.

Underscores Theme per Mausclick: Name des Themes eingeben, auf Generate klicken, fertig.

## 2. Bootstrap einrichten

Im nächsten Schritt lädst Du Bootstrap in der **aktuellen Version (4.3.1) (Compiled CSS and JS)** herunter. Entpacke die Datei, lege sie in das Verzeichnis des gerade erstellten Themes (`/wp-content/themes/meintheme`) und benenne den Ordner in »bootstrap« um. Das FTP-Verzeichnis sieht dann so aus:



## 3. Bootstrap und WordPress verknüpfen

Damit WordPress die JavaScript- und CSS-Dateien von Bootstrap nutzen kann, musst Du nun die `functions.php` des Themes anpassen. Um Zeile 122 befindet sich `function meintheme_scripts()` (statt »meintheme« steht dort der Name Deines Themes). Darunter folgen mehrere Funktionen. Füge vor der schließenden, geschweiften Klammer `}` die folgenden Zeilen ein. Achte darauf, im Code die jeweils gültige Bootstrap-Version anzugeben (in diesem Beispiel 4.1.1):

```
// Bootstrap
wp_register_script( 'bootstrap-js', get_template_directory_uri() . '/bootstrap/js/
bootstrap.min.js', array( 'jquery' ), '4.1.1', true );
wp_register_style( 'bootstrap-css', get_template_directory_uri() . '/bootstrap/
css/bootstrap.min.css', array(), '4.1.1', 'all' );
wp_enqueue_script( 'bootstrap-js' );
wp_enqueue_style( 'bootstrap-css' );
```

In der `functions.php` sieht das wie folgt aus:

```

/**
 * Enqueue scripts and styles.
 */
function meintheme_scripts() {
    wp_enqueue_style( 'meintheme-style', get_stylesheet_uri() );

    wp_enqueue_script( 'meintheme-navigation', get_template_directory_uri() . '/js/navigation.js', array(), '20151215', true );

    wp_enqueue_script( 'meintheme-skip-link-focus-fix', get_template_directory_uri() . '/js/skip-link-focus-fix.js', array(), '20151215', true );

    if ( is_singular() && comments_open() && get_option( 'thread_comments' ) ) {
        wp_enqueue_script( 'comment-reply' );
    }

    // Bootstrap
    wp_register_script( 'bootstrap-js', get_template_directory_uri() . '/bootstrap/js/bootstrap.min.js', array( 'jquery' ), '4.1.1', true );
    wp_register_style( 'bootstrap-css', get_template_directory_uri() . '/bootstrap/css/bootstrap.min.css', array(), '4.1.1', 'all' );
    wp_enqueue_script( 'bootstrap-js' );
    wp_enqueue_style( 'bootstrap-css' );
}
add_action( 'wp_enqueue_scripts', 'meintheme_scripts' );

```

Nach dem Upload ist das Theme mit Bootstrap verknüpft. Damit kannst Du Bootstrap-Elemente in Deinem Blog testen: Einfach die vorgefertigten Code-Blöcke von [getbootstrap.com](https://getbootstrap.com) kopieren und in den WordPress-Editor (Text-Ansicht) einfügen. Nach dem Veröffentlichen erscheinen die Elemente im Frontend Deines Blogs.

## 4. Navigationsleiste einfügen

Um die [Navbar](#) mit Dropdown-Menü in die Menü-Funktion von WordPress zu implementieren, lädst Du zunächst die [class-wp-bootstrap-navwalker.php](#) von der GitHub Respository herunter und legst sie im Theme-Hauptverzeichnis ab. Die `functions.php` muss nun erneut angepasst werden, damit sie die neue Datei aufrufen kann. Füge dazu folgende Zeile ganz ans Ende ein (verwende statt »meintheme« den Namen Deines Themes):

```

// Register Custom Navigation Walker
require_once get_template_directory() . '/class-wp-bootstrap-navwalker.php';

//Register Menu
register_nav_menus( array(
    ,primary' => __( 'Primary Menu', 'meintheme' ),
));

```

Danach löschst Du die aktuelle WordPress-Navigation in der `header.php`, die sich ebenfalls im Theme-Ordner befindet. Entferne den kompletten Code zwischen

```
<header id="masthead" class="site-header"> und <!-- #site-navigation -->.
```

An gleicher Stelle fügst Du diese Zeilen ein:

```
<nav class="navbar navbar-expand-md navbar-light bg-light" role="navigation">
  <div class="container">
    <!-- Brand and toggle get grouped for better mobile display -->
    <button class="navbar-toggler" type="button" data-toggle="collapse"
      data-target="#bs-example-navbar-collapse-1" aria-controls="bs-example-
      navbar-collapse-1" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <a class="navbar-brand" href="<?php echo esc_url( home_url( '/' ) );
    ?>">Katzenfreunde Südsachsen</a>
    <?php
    wp_nav_menu( array(
      'theme_location' => 'primary',
      'depth' => 2,
      'container' => 'div',
      'container_class' => 'collapse navbar-collapse',
      'container_id' => 'bs-example-navbar-collapse-1',
      'menu_class' => 'nav navbar-nav',
      'fallback_cb' => 'WP_Bootstrap_Navwalker::fallback',
      'walker' => new WP_Bootstrap_Navwalker(),
    ));
    ?>
  </div>
</nav>
```

Der Code für den Header sieht anschließend so aus:

```
<header id="masthead" class="site-header">
  <nav class="navbar navbar-expand-md navbar-light bg-light" role="navigation">
    <div class="container">
      <!-- Brand and toggle get grouped for better mobile display -->
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1"
        aria-controls="bs-example-navbar-collapse-1" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <a class="navbar-brand" href="<?php echo esc_url( home_url( '/' ) ); ?>">Katzenfreunde Südsachsen</a>
      <?php
      wp_nav_menu( array(
        'theme_location' => 'primary',
        'depth' => 2,
        'container' => 'div',
        'container_class' => 'collapse navbar-collapse',
        'container_id' => 'bs-example-navbar-collapse-1',
        'menu_class' => 'nav navbar-nav',
        'fallback_cb' => 'WP_Bootstrap_Navwalker::fallback',
        'walker' => new WP_Bootstrap_Navwalker(),
      ));
      ?>
    </div>
  </nav><!-- #site navigation -->
</header><!-- #masthead -->
```

Statt »Katzenfreunde Südsachsen« gibst Du den Namen Deiner Website ein.

## 5. Menü erstellen

Wenn Du es bis hierher geschafft hast, bist Du (fast) am Ziel angekommen! Nachdem Du die aktualisierten Dateien hochgeladen hast, kannst Du im Dashboard unter **Design > Menüs** ein Menü erstellen. Damit ein Dropdown-Menü angezeigt wird, musst Du Unterpunkte anlegen. Setze dafür in den Menü-Einstellungen bei **Primary Menu** ein Häkchen.

### Menü-Struktur

Ziehe die Einträge in deine bevorzugte Reihenfolge. Klicke den Pfeil auf der rechten Seite, um weitere Konfigurations-Optionen anzuzeigen.

- Gesundheit Seite ▼
- Ernährung *Unterpunkt* Seite ▼
- Bewegung & Spiele *Unterpunkt* Kategorie ▼
- Kalzenrassen Seite ▼
- Cat Content Kategorie ▼
- Zuhause gesucht Kategorie ▼

---

### Menü-Einstellungen

Seiten automatisch hinzufügen  Neue Seiten der ersten Ebene automatisch zum Menü hinzufügen

Position im Theme  **Primary Menu**  
 Primary

WordPress-Menü-Einstellungen: Per Drag & Drop kannst Du die Menüpunkte verschieben und so hierarchisch gliedern.

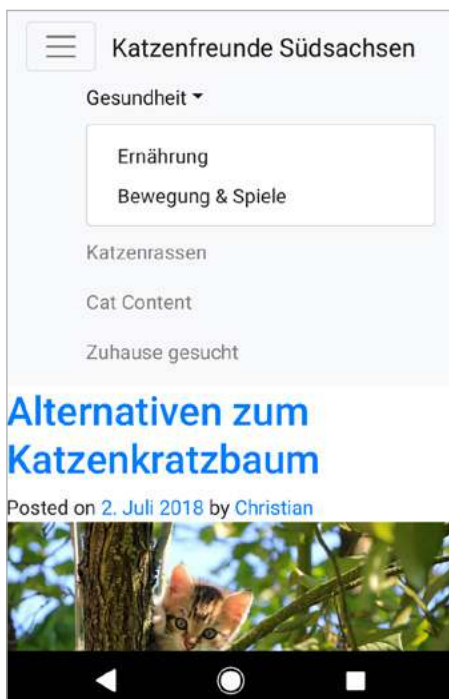


Fertig!

Das Ergebnis Deiner Arbeit kannst Du Dir im Frontend anschauen – ein Bootstrap-Menü im WordPress-Blog:



Desktop-Ansicht



Mobile Ansicht

## 2.9 Eigenes Theme entwickeln: Icons integrieren und Kommentarbereich optimieren

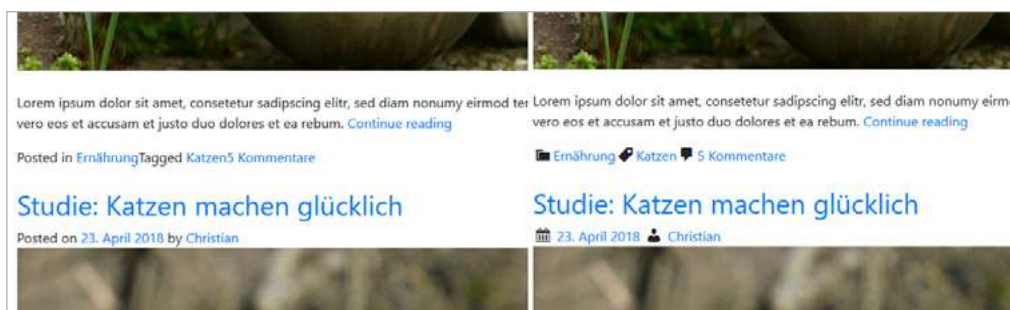
Im vorigen [Kapitel 2.7](#) haben wir gezeigt, wie Du in das Basis-Theme Underscores eine Bootstrap-Navigationsleiste einbaust. Darauf aufbauend erfährst Du nun, wie Du Icons integrierst und Bootstrap-Kommentarfelder verwendest. Mit JavaScript kannst du obendrein die Eingaben Deiner Leser validieren und entsprechende Meldungen einblenden (z. B. »Bitte gib Deine E-Mail-Adresse ein.«).

### Dashicons integrieren

Bootstrap 4 beinhaltet keine Icons. Gut, dass WordPress selbst welche an Bord hat: Die sogenannten [Dashicons](#) sind die Icons, die Du bereits aus dem Backend kennst. Wenn Du folgende Funktion in die `functions.php` eingibst, kannst Du das Icon-Set auch für das Frontend nutzen:

```
// Enqueue Dashicons
function load_dashicons_front_end() {
    wp_enqueue_style( 'dashicons' );
}
add_action( 'wp_enqueue_scripts', 'load_dashicons_front_end' );
```

Um das Theme grafisch aufzulockern, kannst Du nun die Links für Datum, Autor, Kategorien, Tags, Kommentare und Edit (bzw. Bearbeiten) mit passenden Icons versehen. Dazu musst Du die entsprechenden PHP-Dateien anpassen.



Vorher-Nachher: Mit Icons (rechts) wirkt das Theme professioneller.

## Kalender-Icon

Zum Datum passt das Kalender-Dashicon. Trage zu diesem Zweck in die **content.php** (im Hauptverzeichnis) unterhalb von

```
<div class="entry-meta">
```

Folgendes ein (Zeile 23):

```
<span class="dashicons dashicons-calendar"></span>
```

Weil das »Posted on« vor der Datumsangabe durch das Kalender-Icon nun überflüssig ist, kannst Du die Wörter in der Datei **/inc/template-tags.php** löschen (Zeile 29). Die Zeile sieht anschließend so aus:

```
esc_html_x( ,%s', ,post date', ,meintheme' ),
```

## Autor-Icon

Alle folgenden Anpassungen betreffen ebenfalls die **template-tags.php**. Um das User-Icon vor den Autorenamen zu platzieren, fügst Du (mit Leerzeichen am Anfang)

```
<span class="dashicons dashicons-admin-users"></span>
```

in die Zeile 49 ein:

```
echo ,<span class="byline"> <span class="dashicons dashicons-admin-users"></span> . $byline . ,</span>; // WPCS: XSS OK.
```

Das Wort »by« wird nicht mehr benötigt. Lösche es einfach (Zeile 35), sodass die Zeile wie folgt aussieht:

```
esc_html_x( ,%s', ,post author', ,meintheme' ),
```

## Kategorien-Icon

Die weiteren Meta-Angaben werden unterhalb des Inhalts angezeigt. Trotzdem bleibst Du in der gleichen Datei, um auch die Kategorien mit einem Dashicon zu versehen. Die entsprechende Klasse `<span class="dashicons dashicons-category"></span>` fügst Du in die Zeile 56 ein. Außerdem kannst Du wieder überflüssige Wörter streichen (»Posted in«):

```
printf(, <span class="cat-links"><span class="dashicons dashicons-category"></span>' . esc_html__(, %1$s', , meintheme') . , </span>', $categories_list); // WPCS: XSS OK.
```

### Tags-Icon

Wie bei den Kategorien kannst Du überflüssigen Text (»Tagged«) entfernen (Zeile 72). Hier wird die Klasse `<span class="dashicons dashicons-tag"></span>` verwendet:

```
printf(, <span class="cat-links"><span class="dashicons dashicons-category"></span>' . esc_html__(, %1$s', , meintheme') . , </span>', $categories_list); // WPCS: XSS OK.
```

### Kommentare-Icon

In Zeile 77 fügst Du die Klasse `<span class="dashicons dashicons-admin-comments"></span>` hinzu, damit vor dem Kommentare-Link eine Sprechblase angezeigt wird:

```
echo , <span class="comments-link"><span class="dashicons dashicons-admin-comments"></span>';
```

### Edit-Icon


Für ein Edit-Icon ergänzt Du die Zeile 98 um die Klasse `<span class="dashicons dashicons-edit"></span>` :

```
, <span class="edit-link"><span class="dashicons dashicons-edit"></span>',
```

Zum Schluss kannst Du die Außenabstände der Icons über die `style.css` vergrößern:

```
/* Außenabstand Dashicons */
.dashicons {
    margin: 0.2rem;
}
```

## Kommentarbereich mit Bootstrap optimieren

3.  **HTM** sagt:  
[25. Juni 2018 um 11:56 Uhr](#)

Mein Kater macht gerade Diät. Er ist ein bisschen dick geworden, obwohl er sich viel bewegt. Wenn du willst, kann ich dir ein paar Low Carb Rezepte schicken!

[Antworten](#)

**Schreibe einen Kommentar**

Deine E-Mail-Adresse wird nicht veröffentlicht. Erforderliche Felder sind mit \* markiert.

Kommentar

Name \*

E-Mail \*

Website

Meinen Namen, E-Mail und Website in diesem Browser speichern, bis ich wieder kommentiere.

Der Underscores-Kommentarbereich sieht im Auslieferungszustand etwas blass aus.

Der Kommentarbereich ist ein gutes Beispiel für die Verwendung von Bootstrap-Komponenten. Die Eingabefelder von Bootstrap sehen nicht nur besser aus als Underscores-Original, sondern lassen sich per CSS auch einfacher anpassen. Öffne dazu die `comments.php` und lösche diese Zeilen:

```
comment_form();
?>
```

Füge an gleicher Stelle den folgenden Code-Block ein:

```

/* Bootstrap Kommentarfelder */

//$consent
if ( has_action( ,set_comment_cookies', ,wp_set_comment_cookies' ) ) {
    $consent = empty( $commenter[,comment_author_email'] ) ? ,': ,
    checked="checked"";
    $fields[,cookies'] = ,<p class="comment-form-cookies-consent"><input
    id="wp-comment-cookies-consent" name="wp-comment-cookies-consent"
    type="checkbox" value="yes" . $consent . , />' .
    ,<label for="wp-comment-cookies-consent">' . __( ,Save my name, email,
    and website in this browser for the next time I comment.' ) . ,</label></p>;
}

//Bootstrap Klassen
ob_start();
$commenter = wp_get_current_commenter();
$req = true;
$aria_req = ( $req ? „ aria-required='true'" : ,');

$comments_arg = array(
    ,form' => array(
        ,class' => ,form-horizontal'
    ),
    ,fields' => apply_filters( ,comment_form_default_fields', array(
        ,author' => ,<div class="form-group">' . ,<label for="author">'
        . __( ,Name', ,strato' ) . ,</label> , . ( $req ? ,<span>*</span>' : ,') .
        ,<input id="author" name="author" class="form-control" type="text"
        value="" size="30" . $aria_req . , />'
        ,<p id="v1" class="text-danger"></p>' . ,</div>',
        ,email' => ,<div class="form-group">' . ,<label for="email">'
        . __( ,E-Mail', ,strato' ) . ,</label> , . ( $req ? ,<span>*</span>' : ,') .
        ,<input id="email" name="email" class="form-control" type="text" value=""
        size="30" . $aria_req . , />'
        ,<p id="v2" class="text-danger"></p>' . ,</div>',
        ,url' => ,<div class="form-group">' . ,<label for="url">'
        . __( ,Website', ,strato' ) . ,</label> , .
        ,<input id="url" name="url"
        class="form-control" type="text" value="" size="30" />'
        ,<p class="text-danger"></p>' . ,</div>',

```

```

,cookies'          => ,<div class="comment-form-cookies-
                    consent"><input id="wp-comment-cookies-
                    consent" name="wp-comment-cookies-consent"
                    type="checkbox" . $consent . , /> .
                    ,<label for="wp-comment-cookies-consent">' . __( ,Save my name, email,
                    and website in this browser for the next time I comment.' ) . ,</label></
                    div>'),
,comment_field'    => ,<div class="form-group">' . ,<label
                    for="comment">' . __( ,Kommentar', ,strato' ) . ,</
                    label><span>*</span>' .
                    ,<textarea id="comment" class="form-
                    control" name="comment" rows="3" aria-
                    required="true"></textarea>
                    <p id="v3" class="text-danger"></
                    p>' . ,</div>',
,class_submit'     => ,btn btn-primary'
); ?>
<?php comment_form($comments_arg);
echo str_replace(class="comment-form","class="comment-form"
name="commentForm" onsubmit="return validateForm();" ,ob_get_clean());
?>

```

Um zum Antworten einen Bootstrap-Button zu verwenden, fügst Du diese Zeilen an das Ende der **functions.php** ein:

```

// Bootstrap Submit Button
add_action('comment_form', ,bootstrap4_comment_button');
function bootstrap4_comment_button() {
    echo ,<button class="btn btn-primary" type="submit">' . __( ,Kommentar
    abschicken' ) . ,</button>;
}

```

Nach der Aktualisierung erscheint unterhalb des grauen Kommentar-Buttons das blaue Bootstrap-Pendant. Über die **style.css** blendest Du den alten Button einfach aus:

```

/* Underscores-Button ausblenden */
.form-submit {
    display: none;
}

```

Damit auch die Reply-Links als Buttons erscheinen, ersetzt Du in der `functions.php` die originale Klasse durch die Bootstrap-Klasse (in diesem Fall `.btn-primary`):

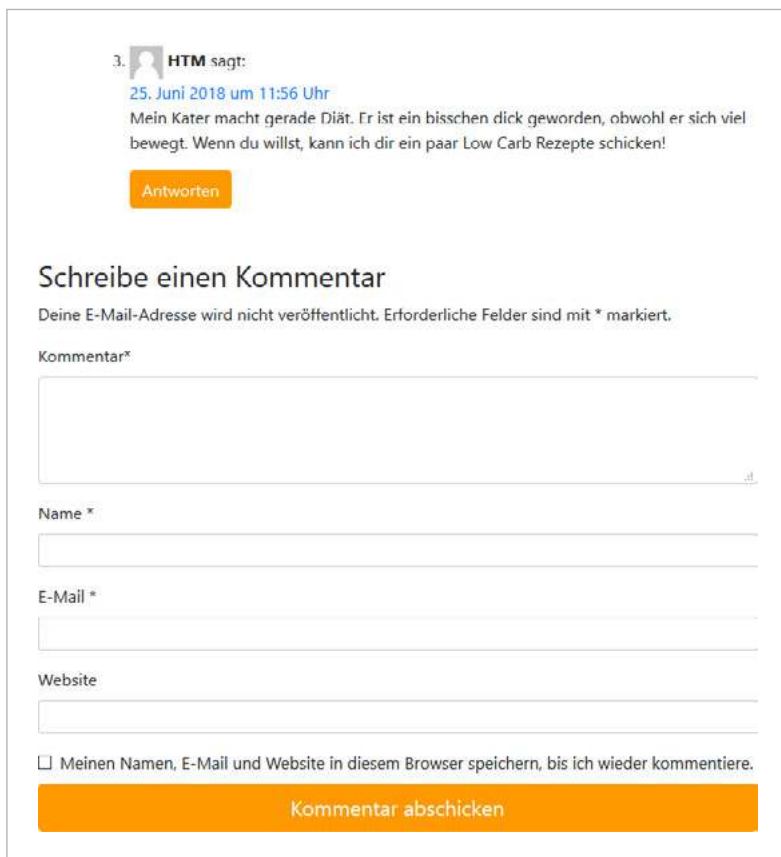
```
// Bootstrap Antworten-Button
add_filter('comment_reply_link', 'replace_reply_link_class');
function replace_reply_link_class($class){
    $class = str_replace('comment-reply-link', 'btn btn-primary',
    $class);
    return $class;
}
```


Über das Stylesheet kannst Du die Abstände und Buttons-Farben anpassen:

```
/* Abstand Reply-Links */
.reply {
    padding-bottom: 2rem;
}
/* Abstand Cookie-Checkbox */
input#wp-comment-cookies-consent {
    margin-right: 0.2rem;
}
/* Farbe Bootstrap-Buttons */
.btn.btn-primary {
    background-color: #ff9900;
    border: #ff9900;
}
```



Wenn alles erledigt ist, sieht Dein Kommentarbereich so aus:



3.  **HTM** sagt:  
25. Juni 2018 um 11:56 Uhr  
Mein Kater macht gerade Diät. Er ist ein bisschen dick geworden, obwohl er sich viel bewegt. Wenn du willst, kann ich dir ein paar Low Carb Rezepte schicken!

[Antworten](#)

### Schreibe einen Kommentar

Deine E-Mail-Adresse wird nicht veröffentlicht. Erforderliche Felder sind mit \* markiert.

Kommentar\*

Name \*

E-Mail \*

Website

Meinen Namen, E-Mail und Website in diesem Browser speichern, bis ich wieder kommentiere.

[Kommentar abschicken](#)

## Validierung der Kommentarfelder

Füllst Du die Pflichtfelder im Kommentarbereich nicht aus, leitet Dich WordPress standardmäßig auf eine neue Seite mit folgender Nachricht weiter:

**FEHLER:** Bitte fülle die erforderlichen Felder aus (Name, E-Mail-Adresse).

[« Zurück](#)

Wäre es nicht besser, auf der Seite zu bleiben und entsprechende Meldungen direkt bei den Kommentarfeldern anzuzeigen?

Falls Du Dir den neuen Code für die **comments.php** genau angeschaut hast, sind Dir vielleicht die IDs v1, v2 und v3 aufgefallen. Damit kannst Du die Kommentarfelder per JavaScript direkt beim Absenden eines Kommentars validieren und individuelle Meldungen anzeigen. Füge dazu diesen Code-Block in die **comments.php** direkt oberhalb von `</div><!-- #comments -->` ein:

```
<script>
  /* basic javascript form validation */
  function validateForm() {
    var form = document.forms[„commentForm“];
    x = form[„author“].value,
    y = form[„email“].value,
    z = form[„comment“].value,
    flag = true,
    v1 = document.getElementById(„v1“),
    v2 = document.getElementById(„v2“),
    v3 = document.getElementById(„v3“);

    if (x == null || x == „“) {
      v1.innerHTML = „<?php echo __(„Namen eingeben“, „strato“); ?>“;
      z = false;
    } else {
      v1.innerHTML = „“;
    }

    if (y == null || y == „“) {
      v2.innerHTML = „<?php echo __(„E-Mailadresse eingeben“, „strato“); ?>“;
      z = false;
    } else {
      v2.innerHTML = „“;
    }

    if (z == null || z == „“) {
      v3.innerHTML = „<?php echo __(„Kommentar eingeben“, „strato“); ?>“;
      z = false;
    } else {
      v3.innerHTML = „“;
    }

    if (z == false) {
      return false;
    }
  }
</script>
```

```
}  
  
}  
</script>
```

Wenn nun ein Besucher die Felder für Kommentar, Name oder E-Mail leer lässt und auf **Kommentar absenden** klickt, erhält er direkt folgende Meldungen:

### Schreibe einen Kommentar

Deine E-Mail-Adresse wird nicht veröffentlicht. Erforderliche Felder sind mit \* markiert.

Kommentar\*

Bitte gib einen Kommentar ein.

Name \*

Bitte gib Deinen Namen ein.

E-Mail \*

Bitte gib Deine E-Mailadresse ein.

Website

Meinen Namen, E-Mail und Website in diesem Browser speichern, bis ich wieder kommentiere.

**Kommentar abschicken**

## 2.10 Eigenes Theme entwickeln: Strukturieren mit HTML und CSS

Im vorigen [Kapitel 2.9](#) haben wir gezeigt, wie Du Icons und Bootstrap-Elemente für die Entwicklung Deines Themes auf Basis von Underscores verwendest. Unabhängig davon, ob es sich um ein eigenes oder vorgefertigtes Theme handelt: Um möglichst flexibel zu sein, hilft es, die HTML-Struktur anzupassen und dann per CSS zu gestalten. Auf diese Weise kannst Du etwa den Content-Bereich mittig platzieren oder eine Sidebar einfügen.

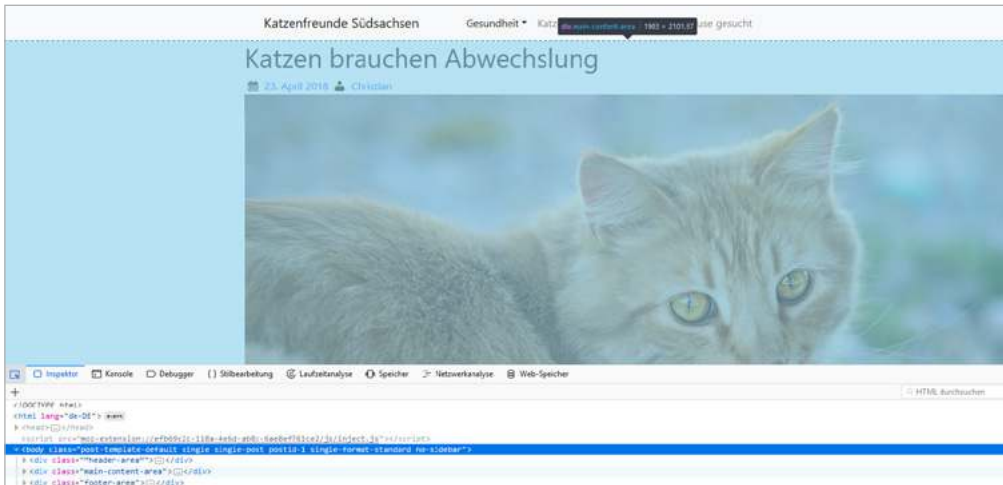
### Flexibel ist besser

Underscores teilt den Inhaltsbereich in **#Primary** (Content) und **#Secondary** (Sidebar). Die Sidebar befindet sich standardmäßig unterhalb des Content-Bereichs, weil dieser die komplette Breite einnimmt. Um den linksbündigen Content-Bereich in die Mitte zu verschieben, könntest Du einfach folgende CSS-Anweisung verwenden:

```
/* Content mittig platzieren */
#page {
  width: 1170px;
  margin: 0 auto;
}
```

Das Problem dabei ist allerdings, dass der Content-Bereich auf die angegebene Breite beschränkt bleibt. Nicht möglich wäre daher beispielweise ein Header, der die volle Bildschirmbreite einnimmt. Darum ist es besser, auf den globalen **#page**-Container zu verzichten und stattdessen separate Klassen einzuführen.

Die Bereiche Header, Content und Footer bekommen je zwei Klassen: Eine eigene Klasse (**area**) für unabhängige CSS-Anpassungen und eine übergeordnete Klasse (**main-page**). Wir passen zu diesem Zweck drei Dateien an, die im Hauptverzeichnis des Themes liegen: **header.php**, **footer.php** und **style.css**.



Die neue Klasse »main-content-area« wird die volle Breite einnehmen.

## HTML-Struktur anpassen

Der Container **#page** beginnt im Header und endet im Footer. Um ihn zu entfernen, löschst Du in der **header.php** die Zeile

```
<div id="page" class="site">
```

und in der **footer.php** folgenden Code:

```
</div><!-- #page -->
```

## Header

Danach fügst Du in der **header.php** an gleicher Stelle eine Klasse für den Header (**.header-area**) und direkt darunter (also innerhalb des Containers) die zweite Klasse (**main-page**) für die gesamte Seite ein:

```
<div class="header-area">
  <div class="main-page">
```

Schließe die beiden Container unmittelbar nach `</header><!-- #masthead -->` mit

```
</div>
</div>
```

## Content

Das Gleiche wiederholst Du für den Content. Füge ebenfalls in der **header.php** oberhalb von

```
<div id="content" class="site-content">
```

Folgendes ein:

```
<div class="main-content-area">  
<div class="main-page">
```

Danach schließt Du die beiden div-tags in der **footer.php** direkt nach

```
</div><!-- #content -->
```

mit

```
</div>  
</div>
```

## Footer

Aller guten Dinge sind drei. Gib nun für den Footer direkt unterhalb des gerade eingegebenen Codes in der **footer.php** dies ein:

```
<div class="footer-area">  
<div class="main-page">
```

Die Tags werden unterhalb von `</footer><!-- #colophon -->` geschlossen mit

```
</div>  
</div>
```

## Content mittig platzieren

Nun verfügt das Theme über je eine Klasse für Header, Content und Footer, die bei Bedarf die volle Breite nutzen können. Mit der übergeordneten Klasse **main-page** kannst Du in der **style.css** eine fixe Breite von beispielsweise 1.170 Pixeln definieren. Die Container **#Primary** und **#Secondary** verwenden unabhängig vom Inhalt immer die maximale Breite. So bleibt die Sidebar bzw. der Widget-Bereich unterhalb des (mittigen) Contents:

```
/* Breite 1170px */
.main-page {
  max-width: 1170px;
  margin: 0 auto;
}

#primary, #secondary {
  float: none;
  padding: 0;
  width: 100%;
}
```

Über **.main-content-area** kannst Du nun den Abstand des Contents zum Footer vergrößern:

```
/* Abstand Content-Footer*/
.main-content-area {
  padding-bottom: 2%;
}
```



Mittig platzierte Inhalte mit einer maximalen Breite von 1.170 Pixeln. Damit die Bilder nicht zu viel Platz einnehmen, ist ein Seitenverhältnis von 16:9 empfehlenswert (1170x658 Pixel).

## Sidebar einfügen

Falls Du in Deinem Fullwidth-Layout doch die Sidebar vermisst, kannst Du problemlos zur bekannten Blog-Aufteilung wechseln. Dabei solltest Du allerdings die unterschiedlichen Auflösungen von Desktops und mobilen Geräten beachten. Die Bootstrap-Navigationsleiste wechselt bei einer Bildschirmbreite unterhalb von 768 Pixeln in den mobilen Modus. An diesem Breakpoint sollte auch die Sidebar nach unten wandern. Ändere dazu die Media Query in der **style.css** wie folgt:

```
/* Sidebar ausblenden (mobil) */
@media screen and (min-width: 768px) {
  .menu-toggle {
    display: none;
  }
  .main-navigation ul {
    display: block;
  }
}
```



```
#primary {
float: left;
padding-right: 3%;
width: 66.5%
}

#secondary {
float: left;
width: 33.5%
}
}
```

Damit sagst Du Deinem Theme, dass das Bootstrap-Menü ab einer Bildschirmbreite von 768 Pixeln nicht mehr eingeklappt wird. Mit den zusätzlichen Anweisungen unten legst Du fest, dass der Content zwei Drittel und die Sidebar ein Drittel der Breite einnehmen.

Katzenfreunde Südsachsen Gesundheit ▾ Katzenrassen Cat Content Zuhause gesucht

## Katzen brauchen Abwechslung

📅 23. April 2018 👤 Christian



Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

### Neueste Beiträge

- Alternativen zum Katzenkratzbaum
- Die 5 beliebtesten Katzenrassen 2018
- Immer mehr Katzen ernähren sich vegan
- Studie: Katzen machen glücklich
- Cindy läuft weg und kommt nach drei Monaten wieder

### Neueste Kommentare

- HTM bei Immer mehr Katzen ernähren sich vegan
- Malte bei Immer mehr Katzen ernähren sich vegan
- Marie bei Immer mehr Katzen ernähren sich vegan
- Christian bei Immer mehr Katzen ernähren sich vegan
- Marie bei Immer mehr Katzen ernähren sich vegan

### Kategorien

- Allgemein
- Bewegung & Spiele
- Cat Content
- Ernährung
- Gesundheit
- Katzenrassen
- Zuhause gesucht

Layout mit Sidebar in der Desktop-Ansicht

Falls Dich die Listenpunkte in den Widgets stören – weg damit:

```
.widget-area ul {
  list-style-type: none;
  margin: 0px;
  padding-left: 0px;
}
```

In der mobilen Ansicht wird das Menü eingeklappt und die Sidebar unterhalb des Inhaltsbereichs platziert:



Der Anfang ist gemacht

Ob Fullwidth- oder klassisches Blog-Layout mit Sidebar: Mit den gezeigten Modifikationen der HTML-Struktur hast Du die Gestaltung Deines Themes selbst in der Hand. Wenn Header oder Footer breiter sein sollen als der Inhaltsbereich, verwendest Du die separaten **area**-Klassen. Farben, Abstände, Schriften kannst Du nach Lust und Laune über die **main-page** ändern.

# 3 Plugins

## 3.1 Dein eigenes WordPress-Plugin leicht gemacht

Bestimmt hast Du auch schon nach Tricks gesucht, um einfache Funktionen in WordPress hinzuzufügen – beispielsweise Shortcodes, Code-Schnipsel zum Integrieren von Google Analytics oder ein Custom Post Type. Meist findest Du dabei Code-Schnipsel, die Du per Copy-and-Paste in die **functions.php** Deines WordPress-Themes übernimmst.

Die **functions.php** zu verändern, hat aber auch Nachteile:

- Änderst Du die **functions.php** direkt im Theme, gehen die Änderungen mit dem nächsten Update verloren. Du solltest dafür also ohnehin immer [ein Child-Theme anlegen](#).
- Wechselst Du eines Tages Dein Theme, musst Du die Änderungen in der **functions.php** mühselig heraussuchen und in das neue Theme übertragen.
- Je mehr Du in der **functions.php** ergänzt, desto unübersichtlicher wird die Datei und desto aufwändiger die Fehlersuche.

### Die Lösung: Dein eigenes Plugin

All diese Probleme vermeidest Du, indem Du die meisten Deiner Anpassungen und Ergänzungen stattdessen in ein eigenes WordPress-Plugin packst. Dort verwaltest Du Code-Schnipsel komfortabel. Wenn sich ein Fehler einschleicht, nimmst Du das Plugin kurzfristig offline und musst nicht am Livesystem in der **functions.php** auf Fehlersuche gehen. Auch in ein neues Theme kannst Du ein solches Plugin mitnehmen.

### Ein Plugin erstellen ist sehr einfach

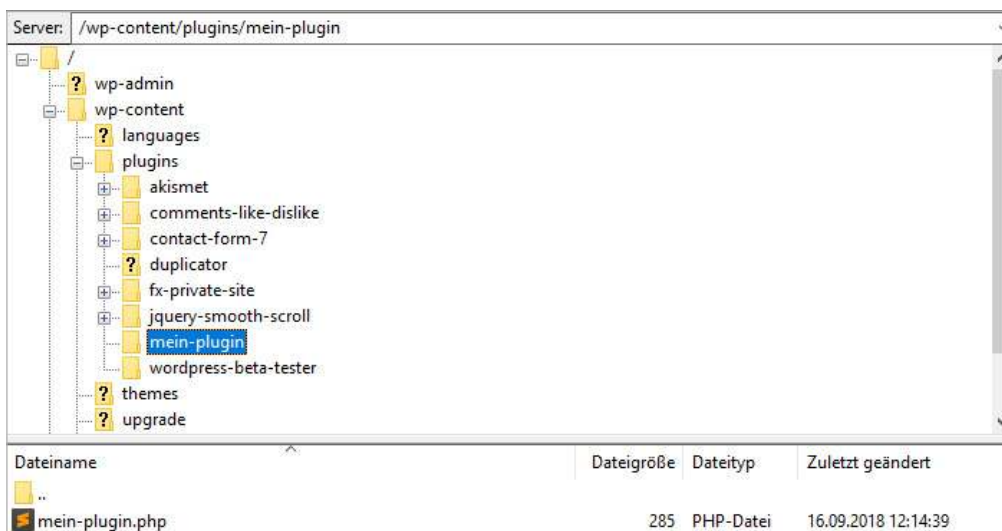
Ein eigenes Plugin zu programmieren, klingt kompliziert, ist es aber nicht. Genau genommen brauchst Du erst einmal nur ein paar Zeilen Text in einer neu anzulegenden PHP-Datei.

```

<?php
defined( 'ABSPATH' ) or die( 'No script kiddies please!' );
/* also read https://codex.wordpress.org/Writing_a_Plugin */
/*
Plugin Name: Name Deines Plugins
Description: kurze Beschreibung
*/
/* Plugin-Code UNTERhalb dieser Zeile */
/* Plugin-Code OBERhalb dieser Zeile */

```

Speichere diese Datei mit einem beliebigen Dateinamen und der Dateierdung **.php** ab, lege im Plugin-Verzeichnis Deiner WordPress-Installation (**/wp-content/plugins**) einen neuen Ordner mit dem gleichen Namen an und lade die Datei dort hinein. Achte darauf, dass der Plugin-Name einzigartig ist und nicht denselben Namen trägt wie ein anderes Plugin.



Dein Plugin gehört in denselben Ordner wie alle anderen WordPress-Plugins.

Anschließend musst Du noch das eben neu erstellte Plugin in WordPress aktivieren. Jetzt hast Du Dein eigenes WordPress-Plugin – allerdings noch ohne Funktion.

## Eigene Funktionalität hinzufügen

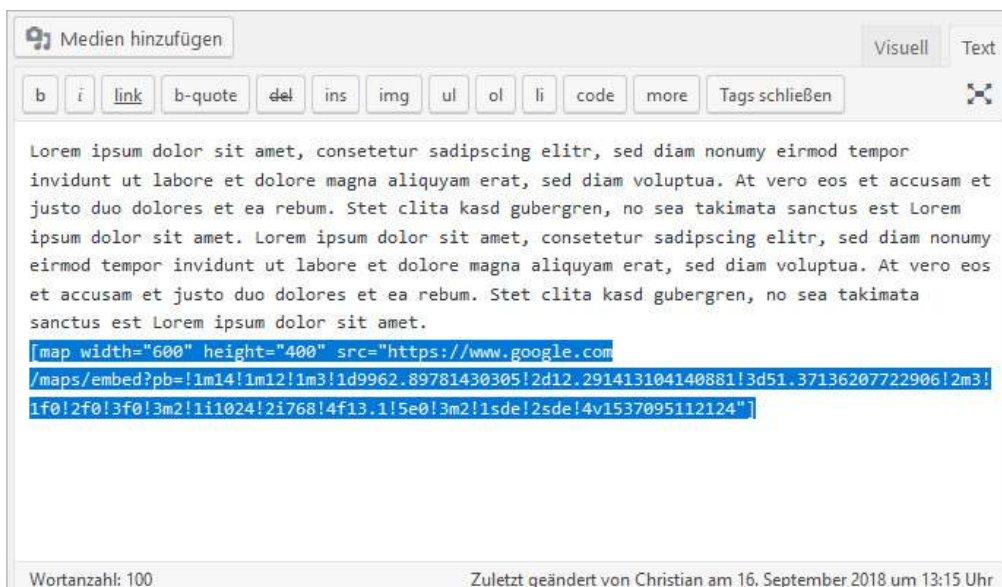
Von nun an fügst Du neue Code-Schnipsel in dieses Plugin ein, statt sie in die **functions.php** Deines Themes zu kopieren. Gegebenenfalls kannst Du natürlich bereits früher eingefügten Code aus der **functions.php** hierher umziehen.

Die Möglichkeiten sind vielfältig, von kurzen Code-Schnipseln mit nützlichen Detailfunktionen über Shortcodes bis hin zu Custom Post Types – also alles, was Du sonst auch in der `functions.php` tun kannst.

Um zu demonstrieren, wie Dein Plugin zum Leben erwacht, definieren wir beispielhaft einen neuen Shortcode. Wir möchten Google-Landkarten in Postings einbauen und deren Breite und Höhe jeweils individuell definieren. Der Shortcode dazu soll so aussehen:

```
[map width="600" height="400" src="https://www.google.com/maps/embed?..."]
```

An die Stelle von »`https://www.google.com/maps/embed?...`« gibst Du die URL ein, die Du bei Google Maps (Burger-Menü > **Karte teilen oder einbetten** > **Karten einbetten**) abrufst. Als Source (src) fügst Du dabei nur die Karten-URL (https) ein (ohne `iframe`, `width`, `height` etc.).



Über den Shortcode definierst Du die Breite, Höhe und Karten-URL.

Das Plugin mit dem dazugehörigen PHP-Code sieht dann so aus:

```

/* Shortcode [map width="" height="" src=""] - Google-Karte einfüegen */
function MeineGoogleMap($atts) {
    extract(shortcode_atts(array(
        „width“ => ,800‘,
        „height“ => ,600‘,
        „src“ => ‘
    ), $atts));
    return ,<iframe width=“.$width.“ height=“.$height.“ frameborder=“0“
    scrolling=“no“ marginheight=“0“ marginwidth=“0“ src=“.$src.“></iframe>;
}
add_shortcode(„map“, „MeineGoogleMap“);

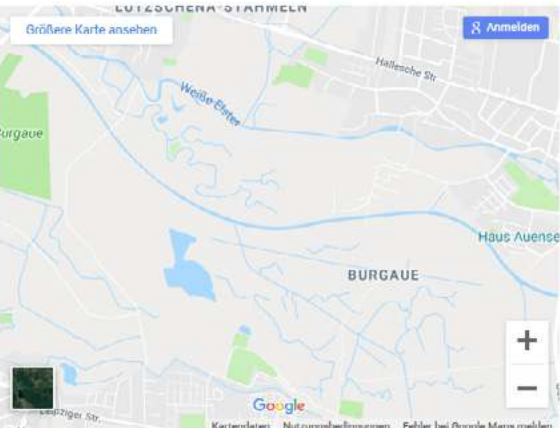
```

Die Shortcode-Funktion in Deinem Plugin erledigt den Rest:

**2. JULI 2018 VON CHRISTIAN**

## Alternativen zum Katzenkratzbaum

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.



**NEUESTE BEITRÄGE**

- [Alternativen zum Katzenkratzbaum](#)
- [Die 5 beliebtesten Katzenrassen 2018](#)
- [Immer mehr Katzen ernähren sich vegan](#)
- [Studie: Katzen machen glücklich](#)
- [Cindy läuft weg und kommt nach drei Monaten wieder](#)

**NEUESTE KOMMENTARE**

- HTM bei Immer mehr Katzen ernähren sich vegan
- Malte bei Immer mehr Katzen ernähren sich vegan
- Marie bei Immer mehr Katzen ernähren sich vegan
- Christian bei Immer mehr Katzen ernähren sich vegan
- Marie bei Immer mehr Katzen ernähren sich vegan

**KATEGORIEN**

- [Allgemein](#)

## Welcher Code gehört ins Plugin, welcher nicht?

Gut aufgehoben ist in Deinem Plugin sämtlicher Code, der nichts direkt mit Deinem Theme zu tun hat. Das sind also Funktionen, die auch unabhängig vom Theme nützlich sind.

Im Umkehrschluss solltest Du alles, was direkten Einfluss auf das Theme nimmt, auf dessen Funktionen aufbaut oder solche verändert, in der **functions.php** belassen. Ein typisches Beispiel dafür wäre die Definition zusätzlicher Bildgrößen für die Medien-Bibliothek.

Denn: Wechselst Du später einmal das Theme, wären diese Code-Teile im Plugin heimatlos und würden WordPress mit unnötigem, weil dann nutzlosem Code aufblähen.

## Tipp: Strukturiere Dein Plugin mit Kommentarzeilen

Gewöhnlich enthält ein Plugin genau eine Funktionalität. Im Falle Deines individuellen Plugins packst Du aber viele voneinander unabhängige Code-Schnipsel in ein einziges Plugin. Damit Du Dich später darin noch zurechtfindest, solltest Du Ordnung halten und alles gut dokumentieren – mit Kommentaren und auch außerhalb des Plugins, um dessen Code nicht aufzublähen.

Gib innerhalb des Plugins jeder Funktion per PHP-Kommentar **/\* kommentar \*/** einen Namen oder eine Kurzbeschreibung und lege Dir dann separat eine Textdatei an, in die Du Details zur jeweiligen Funktionalität hinterlegst. Besonders wichtig ist dabei die Quelle, aus der Du den Code-Schnipsel bezogen hast. Denn dann kannst Du später bei eventuell auftretenden Problemen dort nachsehen und eventuelle Updates finden.

## 3.2 Plugins ersetzen durch Code am Beispiel »Related Posts«

Mit sogenannten »Related Posts«-Plugins kannst Du bequem weitere Beiträge mit gleichen Schlagworten anzeigen. Plugins müssen allerdings aktualisiert werden, sind potenzielle Sicherheitslücken und können Dein Blog ausbremsen. Es geht aber auch anders – ohne Plugin und Nebenwirkungen.

Um nach dieser Anleitung Beiträge unterhalb eines Beitrags mit Bild anzuzeigen, musst Du diese mit einem Beitragsbild und (identischen) Schlagworten versehen.

### Ähnliche Beiträge anzeigen

Zunächst legst Du fest, wo die ähnlichen Beiträge angezeigt werden sollen. Dafür bietet sich etwa der Bereich zwischen Beitrag und Kommentarbereich an. Wo der Code genau eingefügt wird, hängt vom verwendeten Theme ab. Beim Theme Twenty Seventeen zum Beispiel fügst Du die Zeilen in die **single.php** oberhalb des Kommentars »„// If comments are open or we have at least one comment, load up the comment template.« ein.

```
// Ähnliche Beiträge anzeigen
$orig_post = $post;
global $post;
$tags = wp_get_post_tags($post->ID);
if ($tags) {
    $tag_ids = array();
    foreach($tags as $individual_tag) $tag_ids[] = $individual_tag->term_id;
    $args=array(
        ,tag__in' => $tag_ids,
        ,post__not_in' => array($post->ID),
        ,date_query' => array( array( ,after' => ,-1 year' ) ), // Nur Beiträge jünger als ein Jahr
        ,posts_per_page'=>3, // Anzahl der angezeigten Beiträge
        ,caller_get_posts'=>1,
        ,orderby'=>'rand'
    );
    $my_query = new wp_query( $args );
    if( $my_query->have_posts() ) {

echo ,<div id="related-posts"><h3><b>Ähnliche Beiträge</b></h3><ul>;
```



```

while( $my_query->have_posts() ) {
$my_query->the_post();?>

<li><div class="related-images"><a href="<? the_permalink()?>" rel="bookmark"
title="<?php the_title();?>"><?php the_post_thumbnail();?></a></div>
<div class="related-content">
<h3><a href="<? the_permalink()?>" rel="bookmark" title="<?php the_title();
?>"><?php the_title();?></a></h3>
</div>
</li>
<? }
echo ,</ul></div>;
}
}
$post = $orig_post;
wp_reset_query();

```

Beim Theme Twenty Seventeen sieht das im Editor so aus:

```

<?php
/* Start the loop */
while ( have_posts() ) : the_post();

    get_template_part( 'template-parts/post/content', get_post_format() );

    // Ähnliche Beiträge anzeigen
    $orig_post = $post;
    global $post;
    $tags = wp_get_post_tags($post->ID);
    if ($tags) {
        $tag_ids = array();
        foreach($tags as $individual_tag) $tag_ids[] = $individual_tag->term_id;
        $args = array(
            'tag_in' => $tag_ids,
            'post_not_in' => array($post->ID),
            'date_query' => array( array( 'after' => '-1 year' ) ), // Nur Beiträge jünger als ein Jahr
            'posts_per_page' => 3, // Anzahl der angezeigten Beiträge
            'caller_get_posts' => 1,
            'orderby' => 'rand'
        );
        $my_query = new WP_Query( $args );
        if( $my_query->have_posts() ) {

            echo '<div id="related-posts"><h3><b>Ähnliche Beiträge</b></h3><ul>';

            while( $my_query->have_posts() ) {
                $my_query->the_post();?>

                <li><div class="related-images"><a href="<? the_permalink()?>" rel="bookmark" title="<?php the_title();?>"><?php the_post_thumbnail();?></a></div>
                <div class="related-content">
                <h3><a href="<? the_permalink()?>" rel="bookmark" title="<?php the_title();?>"><?php the_title();?></a></h3>
                </div>
                </li>
            }
            <? }
            echo '</ul></div>';
        }
    }
    $post = $orig_post;
    wp_reset_query();

    // If comments are open or we have at least one comment, load up the comment template.

```

Die mit »//Nur Beiträge jünger als ein Jahr« und »//Anzahl der angezeigten Beiträge« kommentierten Zeilen sind selbsterklärend. Auf diese Weise werden drei ähnliche Beiträge angezeigt, die nicht älter als ein Jahr sind. Um Leser nicht abzulenken, sind weniger Beiträge besser als mehr.

Unterhalb der Artikel werden nun Beiträge in zufälliger Reihenfolge angezeigt, die mindestens ein identisches Schlagwort haben. In diesem Beispiel sind das Beiträge mit dem Schlagwort »Katzen«. Das Beitragsbild und der Titel werden automatisch verlinkt:

# KATZEN

---

### Ähnliche Beiträge



- [Studie: Katzen machen glücklich](#)

**ARCHIVE**

---

[April 2018](#)

---

**KATEGORIEN**

---

[Allgemein](#)

---

**META**

---

[Anmelden](#)

---

[Beitrags-Feed \(RSS\)](#)

---

[Kommentare als RSS](#)

---

[WordPress.org](#)

## Darstellung anpassen

Die Bilder fügen sich zwar in das bestehende Layout ein. Noch stören aber die Listenpunkte links neben den Bildern und die Abstände passen noch nicht. Mit ein wenig Feinjustierung per CSS ist auch das kein Problem. Den Code fügst Du einfach in die `style.css` Deines Child-Theme ein:

```
/* Ähnliche Beiträge */
#related-posts {
    margin-top: 15px;
}
#related-posts h3 {
    padding: 0px 0px 15px;
}
#related-posts li {
    list-style: none !important;
}
```

Deine Anpassungen solltest Du auf dem Desktop und dem Smartphone bzw. mit den [Entwickler-Werkzeugen Deines Browsers](#) testen. Mobil sehen die ähnlichen Beiträge bei uns damit so aus:

# KATZEN

---

**Ähnliche Beiträge**



Kater mag keine Mäuse fangen



Wahl der Miss Katze 2018



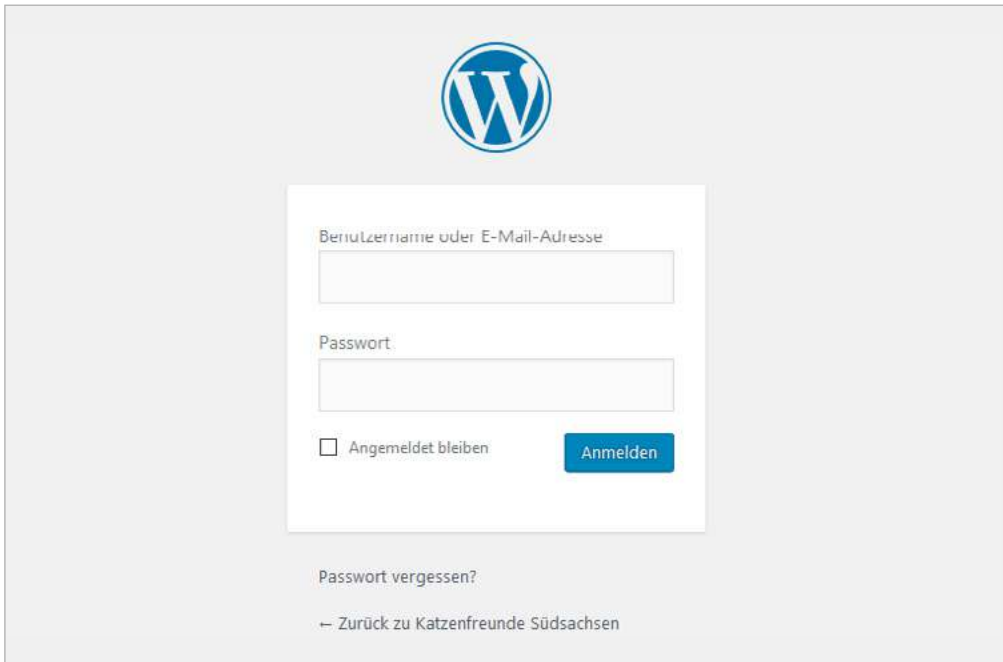
Cindy läuft weg und kommt nach drei Monaten wieder

## Selbermachen oder »Related Posts«-Plugin?

Vielleicht fragst Du Dich an dieser Stelle, ob sich der ganze Aufwand überhaupt lohnt. Kommt drauf an: Wenn die Implementierung möglichst schnell und einfach sein soll, ist die Installation eines gepflegten Plugins der bessere Weg. Selbermachen ist dann angesagt, wenn Du Dein WordPress schlanker machen und selbst bestimmen willst, welche Beiträge angezeigt werden sollen. Weitere Vorteile sind: Mit dem Plugin-freien Vorgehen wirst Du unabhängiger von Dritten, reduzierst Risiken und musst Dir weniger Gedanken über Updates machen. Und letztlich macht das manuelle Einrichten oft mehr Spaß als die Installation auf Knopfdruck.

# 4 Backend

## 4.1 WordPress-Login bunter machen



Standard-Login von WordPress

Grauer Hintergrund, WordPress-Logo und schlichtes Formular müssen nicht sein. Schließlich kannst Du neben [›Theme](#) und [›Backend](#) auch den Login-Bereich von WordPress an Deine Bedürfnisse anpassen. Du hast beispielsweise die Möglichkeit, ein Hintergrundbild einzufügen, ein eigenes Logo zu nutzen und das Formular anzupassen.

### Wie kann ich den Login-Bereich anpassen?

Für die Anpassungen benötigst Du zunächst eine CSS-Datei, die Du am besten in einem Unterordner Deines Child-Themes ablegst. In unserem Beispiel legen wir dazu das Verzeichnis »login« an und speichern darin die Datei **login.css**. Der Pfad sieht beim Child-Theme von Twenty Seventeen dann so aus: `/wp-content/themes/twentyseventeen-child/login/login.css`.



Die Datei login.css wird im Child-Theme abgelegt.

Damit das Theme die nachfolgenden Änderungen übernimmt, gibst Du anschließend einfach folgende Zeilen ans Ende der **functions.php** Deines Child-Themes ein:

```
function strato_login() {
    echo <link rel="stylesheet" type="text/css" href="" . get_bloginfo(,stylesheet_
    directory') . ./login/login.css" />;
}
add_action(,login_head', ,strato_login');
```

## Hintergrundbild einfügen und Farben ändern

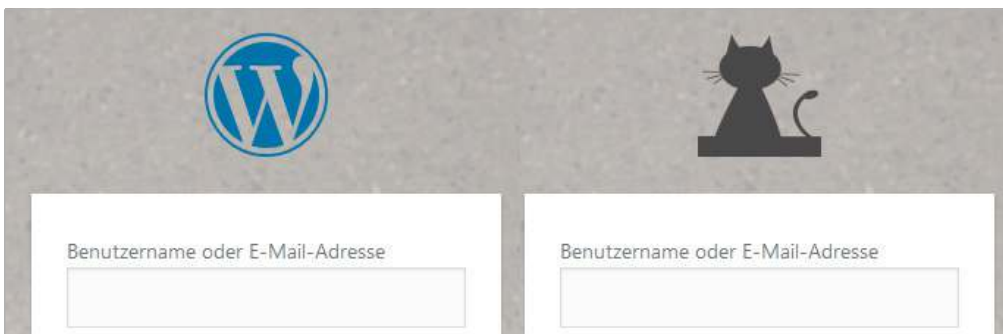
Nachdem das Child-Theme vorbereitet ist, kannst Du in die **login.css** beliebige Anweisungen eingeben. Falls Du ein Hintergrundbild verwenden möchtest, lädst Du das am besten in den Ordner »login«. Mit diesem Code fügen wir zum Beispiel ein Hintergrundbild ein und färben die Links unterhalb des Formulars weiß:

```
body.login {
  background: url(/wp-content/themes/twentyseventeen-child/login/login-
  hintergrund.jpg');
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-position: center;
}
.login #backtoblog a, .login #nav a {
  color: #ffffff;
}
```

## WP Logo durch Katzenbild ersetzen

Das sieht schon besser aus, es fehlt aber noch das Katzenbild (oder Logo). Dazu lädst Du eine 84x84 Pixel große Datei namens **login-katze.png** in den login-Ordner. Mit folgenden CSS-Zeilen bindest Du das Katzenbild ein:

```
#login h1 a {
  background-image: url(/wp-content/themes/twentyseventeen-child/login/
  login-katze.png');
}
```



Login-Bereich mit WordPress-Logo (links) und Katzenbild

## Eingabefelder des Formulars abrunden

Darüber hinaus kannst Du auch das Formular bequem selbst anpassen: Du hättest das Formular und die Eingabefelder gern abgerundet statt eckig? Die Felder mit blauen statt grauen Rahmen? Kein Problem mit diesem Code:

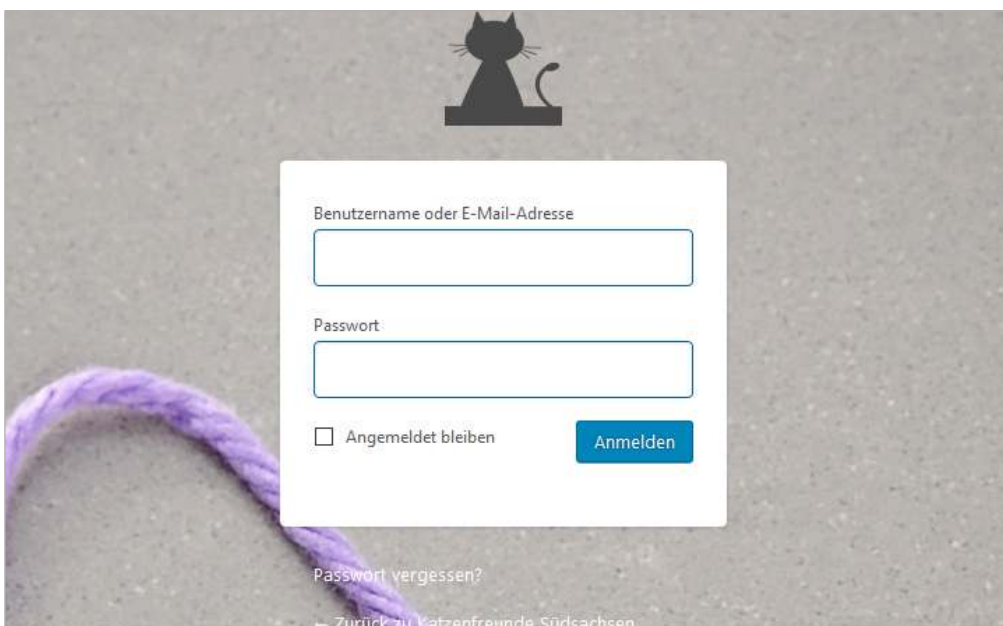
```
.login label {
  font-size: 12px;
  color: #555555;
}

.login input[type="text"]{
  background-color: #ffffff;
  border-color:#0164A5;
  -webkit-border-radius: 4px;
}

.login input[type="password"]{
  background-color: #ffffff;
  border-color:#0164A5;
  -webkit-border-radius: 4px;
}

.login form {
  -webkit-border-radius: 4px;
}
```

Das Ergebnis ist ein WordPress-Login mit Hintergrundbild, Katze und einem schöneren Formular:





## Login kann auch bunt sein

Es ist so einfach, den Standard-Login von WordPress aufzuwerten. Mit wenigen Handgriffen schaffst Du eine Login-Seite, die sich im wahrsten Sinne des Wortes von der grauen Masse abhebt. Zusammen mit den [Backend-Anpassungen](#) fühlt sich WordPress damit viel mehr wie etwas Eigenes, Individuelles an.

## 4.2 Backend anpassen: Individuell statt von der Stange

Keine Lust mehr auf das Standard-Grau? Neben dem Theme (Frontend) lässt sich auch das Backend von WordPress flexibel individualisieren. So kannst Du beispielsweise Farben und Menüpunkte ändern und ein eigenes Logo einfügen. Das wirkt nicht nur professioneller, sondern beschleunigt auch Deinen Workflow.

Bitte vorher ein [Backup](#) aller WordPress-Dateien machen, damit die Website im Notfall wiederhergestellt werden kann! Die genannten PHP-Code-Beispiele fügst Du einfach ans Ende der functions.php des [Child-Themes](#) ein.

### Farben ändern

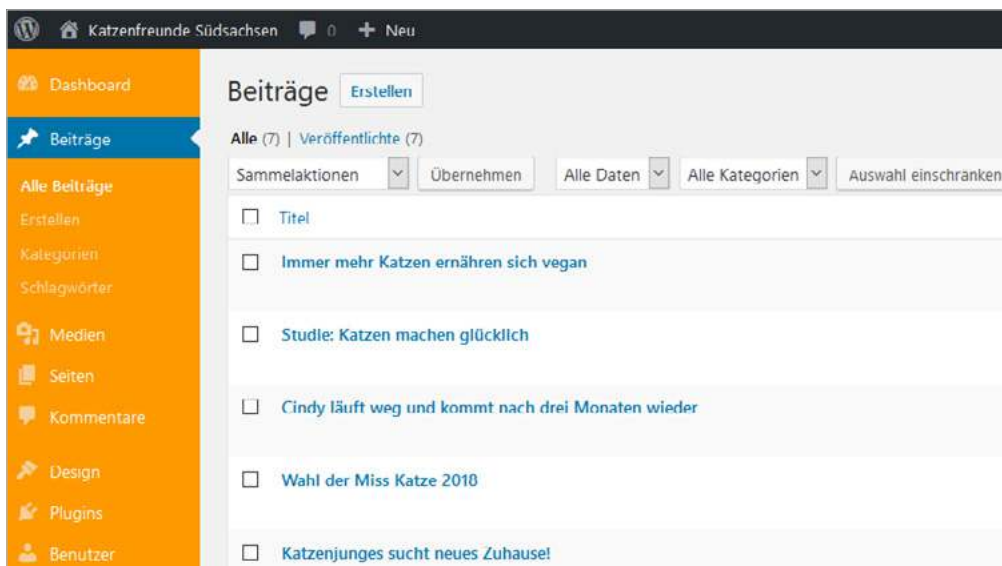
Das grau-schwarz-blaue Standard-Farbschema des WordPress-Backends ist funktional, aber auch Geschmackssache. Über **Benutzer > Dein Profil** stehen zwar alternative Vorlagen zur Auswahl, mit einer eigenen CSS-Datei für das Backend bist Du aber deutlich flexibler. Diese wird mit folgenden Zeilen in das Theme eingebunden:

```
function admin_style() {  
    wp_enqueue_style('admin-styles', get_stylesheet_directory_uri().'/admin.css');  
}  
add_action('admin_enqueue_scripts', 'admin_style');
```

In diesem Beispiel nennen wir die CSS-Datei **admin.css** und legen sie im Hauptverzeichnis des Child-Themes ab. Mithilfe des folgenden CSS-Codes färbst Du das Admin-Menü auf der rechten Seite und die Untermenüs orange:

```
#adminmenu, #adminmenu .wp-submenu, #adminmenuback, #adminmenuwrap {
  background-color: #ff9900 !important;
}
#adminmenu div.wp-menu-name {
  color: #ffffff;
}
```

So sieht das Ergebnis aus:



## Menü ausblenden

Das Menü kannst Du ebenfalls anpassen. Benötigst Du bestimmte Punkte nicht, kannst Du sie nach dem Muster `remove_menu_page( ,XXX.php' );` einfach ausblenden. Bitte beachte, dass das Menü dabei aktiv bleibt und die Funktionen weiterhin über die URL aufrufbar sind. Um Zugriffsrechte für bestimmte Benutzer zu beschränken, solltest Du die entsprechenden Rollen von WordPress nutzen (also statt Administrator z. B. Autor oder Mitarbeiter).

Die nächsten Code-Zeilen dienen der Übersicht, welche Funktionen Du wie ausblendest. Anhand der Kommentierung siehst Du, welche Zeile für welchen Bereich zuständig ist. Würdest Du den kompletten Code übernehmen, wäre Deine Menüleiste leer.

```
function strato_remove_menus(){
    remove_menu_page( ,index.php' );           //Dashboard
    remove_menu_page( ,edit.php' );           //Beiträge
    remove_menu_page( ,upload.php' );         //Medien
    remove_menu_page( ,edit.php?post_type=page' ); //Seiten
    remove_menu_page( ,edit-comments.php' ); //Kommentare
    remove_menu_page( ,themes.php' );         //Design
    remove_menu_page( ,plugins.php' );        //Plugins
    remove_menu_page( ,users.php' );          //Benutzer
    remove_menu_page( ,tools.php' );          //Werkzeuge
    remove_menu_page( ,options-general.php' ); //Einstellungen
}
add_action( ,admin_menu' , ,strato_remove_menus' );
```

Möchtest Du beispielsweise **Plugins** und **Einstellungen** ausblenden, gibst Du folgenden Code ein:

```
function strato_remove_menus(){
    remove_menu_page( ,plugins.php' );        //Plugins
    remove_menu_page( ,options-general.php' ); //Einstellungen
}
add_action( ,admin_menu' , ,strato_remove_menus' );
```

## Menüs umbenennen

Du kannst die Menüpunkte nicht nur ausblenden, sondern auch umbenennen. Um etwa »Beiträge« durch einen anderen Begriff zu ersetzen, gehst Du folgendermaßen vor:

```
function strato_post_menu_rename() {
    global $menu;
    $menu[5][0] = ,News';
}
add_action( ,admin_menu' , ,strato_post_menu_rename' );
```

In diesem Fall nennen wir das Menü »Beiträge« in »News« um.

WordPress ordnet jedem Menü einen Key zu, der hier angegeben werden muss. Das Menü »Beiträge« hat den Key 5, danach folgt Key 0 für die **index.php** (siehe Zeile 3). Auf diese Weise kannst Du sämtliche Menüpunkte im WordPress-Backend umbenennen. Eine Übersicht über die benötigten Keys findest Du [hier](#).

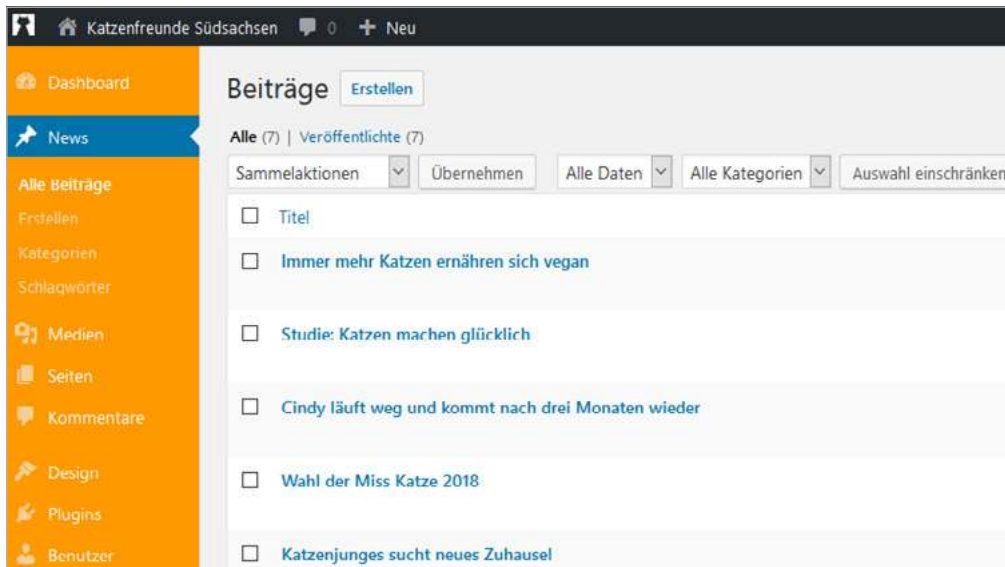
## Katzen-Logo einfügen

Es gibt verschiedene Wege, ein Logo im Backend anzuzeigen. Der einfachste besteht darin, das kleine WordPress-Logo in der Ecke oben links auszutauschen. Dazu legst Du eine 18×18 Pixel große Datei als **backend-katze.png** im Hauptverzeichnis Deines Child-Themes ab und ergänzt die **functions.php** um folgende Zeilen:

```
function wpb_custom_logo() {
    echo ,
    <style type="text/css">
    #wpadminbar #wp-admin-bar-wp-logo > .ab-item .ab-icon:before {
    background-image: url(, . get_bloginfo(,stylesheet_directory') . ./backend-katze.
    png) !important;
    background-position: 0 0;
    background-repeat: no-repeat;
    color:rgba(0, 0, 0, 0);
    }
    #wpadminbar #wp-admin-bar-wp-logo.hover > .ab-item .ab-icon {
    background-position: 0 0;
    }
    </style>
    ;
}

add_action('wp_before_admin_bar_render', ,wpb_custom_logo');
```

Danach sieht das Backend so aus:



Ein gutes Backend ist intuitiv

Layout-Anpassungen müssen nicht auf das Frontend beschränkt bleiben. Mit den gezeigten Anpassungen profitierst Du von einer Oberfläche, die an Deine Bedürfnisse angepasst ist. Das individuelle Backend kannst Du intuitiver bedienen und es sieht auch noch besser aus.

# 5 Sicherheit

## 5.1 Strategischer Hintergrund: Welche Sicherheitslücken solltest Du schließen?

Um Dein Blog vor Hackern zu schützen, bedarf es mehr als ein paar Sicherheitsmaßnahmen in WordPress selbst. Nur mit einer Sicherheits-Strategie, die Deine sämtlichen Online-Aktivitäten umfasst, bist Du wirklich geschützt.

Wenn ein Angreifer sich auf anderem Weg beispielsweise Dein Admin-Passwort besorgen kann oder das WordPress-Login einfach umgeht, helfen alle WordPress-Sicherheitsmaßnahmen nichts. Deshalb ist es noch wichtiger, sich eine Sicherheitsstrategie zuzulegen, die auch das gesamte Umfeld absichert: ein FTP-Zugang, der Login zum STRATO-Backend, der eigene E-Mail-Account und einiges mehr.

Aber ist das nicht übertrieben? Schließlich hast Du nichts für Hacker Interessantes – denkst Du. Was Hacker jedoch tatsächlich suchen, sind Ressourcen, die sie für illegale Zwecke missbrauchen können, ohne dabei entdeckt zu werden: Sie wollen zum Beispiel illegale Dateien, Bilder und Videos tauschen oder schädlichen Code verbreiten. Um ihr Ziel zu erreichen, platzieren sie außerdem unbemerkt Spam-Links oder tauschen Werbebanner und Affiliate-Codes aus, sodass nicht mehr Du, sondern der Hacker die Vermittlungsprovisionen kassiert.

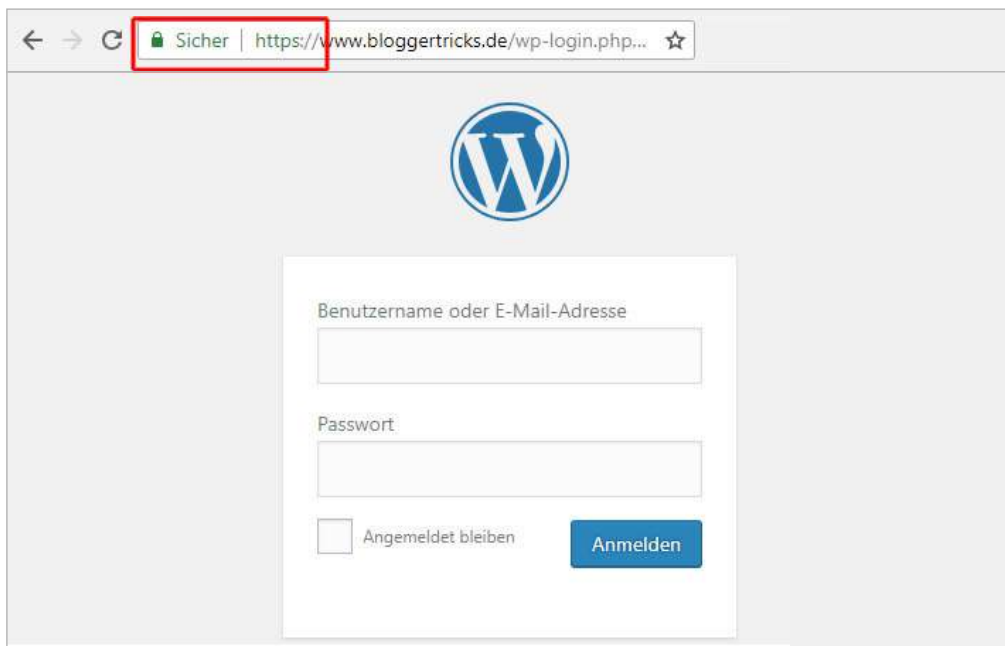
Hacker-Angriffe geschehen automatisiert und systematisch. Fällt einem Hacker beispielsweise Dein E-Mail-Passwort in die Hände, weil Dein E-Mail-Provider gehackt wurde, oder filtert er in einem offenen WLAN Dein FTP-Passwort heraus, kann er mit diesen Informationen unter Umständen kompletten Zugriff auf Deinen Webespace bekommen.

Deshalb ist es wichtig, dass Du Dich generell möglichst sicher durchs Internet bewegst und keine einfachen Angriffspunkte bietest. Einige der nachfolgend aufgezeigten Maßnahmen sind sehr einfach umzusetzen, andere bedeuten etwas Aufwand oder die Umstellung Deiner Gewohnheiten.

### Verschlüsselte Verbindungen

Greife auf sicherheitsrelevante Websites wie beispielsweise WordPress, Dein Webespace per FTP, Deinen E-Mail-Account oder Dein Onlinebanking nur über sichere Verbindungen zu. Das kann am Handy eine Datenverbindung Deines Mobilfunkanbieters sein oder Dein eigener Internetzugang zu Hause.

Bist Du in fremden WLANs unterwegs, sind zusätzliche Sicherheitsmaßnahmen unabdingbar. Grundsätzlich lassen sich die Daten in einem offenen WLAN von jedem mitlesen, der ebenfalls mit diesem WLAN verbunden ist.



Der Login bei WordPress findet sinnvollerweise nur über eine SSL-Verbindung statt

Mindestens solltest Du also Passwörter nur über gesicherte Verbindungen übertragen, also SSL-verschlüsselt. Bei den meisten E-Mail-Anbietern ist das der Fall, für Dein eigenes WordPress musst Du die [SSL-Verschlüsselung](#) aber selbst einrichten.

Solltest Du dennoch einmal ein Passwort über eine unverschlüsselte Verbindung nutzen müssen, ändere das Passwort hinterher so bald wie möglich über eine sichere Verbindung.

## VPN-Verbindung nutzen

Deutlich sicherer fährst Du mit einer sogenannten VPN-Verbindung. Dabei stellt Dein Laptop oder Smartphone eine verschlüsselte Verbindung zu einem Server des VPN-Dienstleisters her. Er überträgt alle Daten komplett verschlüsselt und damit für Angreifer quasi unsichtbar zunächst auf diesen Server. Erst von dort aus gehen die Daten dann an den eigentlichen Bestimmungsort, also beispielsweise zu Deinem

WordPress-Login. Ein Hacker kann so – anders als bei einer SSL-Verschlüsselung – nicht einmal sehen, was Du überhaupt im Internet machst.

VPN-Dienste gibt es in gewissem Umfang kostenlos, beispielsweise das renommierte, in der Schweiz ansässige [Proton VPN](#). VPNs mit schneller Datenübertragung und unbegrenztem Volumen kosten allerdings monatliche Gebühren ab etwa 5 Euro. Auch manche Desktop-Firewalls bringen inzwischen eigene VPN-Funktionen mit. Wer zu Hause mit einer Fritzbox surft, kann sich von unterwegs [per VPN mit der Fritzbox verbinden](#) und auf diesem Weg die sichere heimische Internetverbindung nutzen.

## Sichere Passwörter

Eigentlich selbstverständlich, aber aus Bequemlichkeit oft vernachlässigt: Nutze für alle Accounts [wirklich sichere Passwörter](#) und verwende für jeden Zugang ein anderes Passwort. Denn nutzt Du identische Passwörter beispielsweise für die WordPress-Datenbank, Deinen STRATO Zugang, den FTP-Zugriff auf Deinen Webservice oder den E-Mail-Account, vervielfacht sich für einen Hacker die Chance, dieses eine Passwort zu knacken und damit Zugriff auf all Deine Accounts zu bekommen.

Auch wenn Dein E-Mail-Account auf den ersten Blick nichts mit WordPress zu tun hat: Über die WordPress-Funktion zum Wiederherstellen des Passwortes ist es ein Leichtes, über Deinen E-Mail-Account die Kontrolle über Deine WordPress-Installation zu übernehmen.

## FTP: nur verschlüsselt

Besonders leicht übersieht man die Risiken bei der Datenübertragung per FTP. Gängige FTP-Clients wie Filezilla übertragen Daten nämlich standardmäßig mit dem unverschlüsselten FTP-Protokoll und speichern FTP-Passwörter unverschlüsselt auf der Festplatte.

In Filezilla kannst Du SFTP (SSH File Transfer Protocol) über die Verbindungseinstellungen aktivieren. Als Server legst Du dort `ssh.strato.de` fest, als Protokoll wählst Du **SFTP**. Der Benutzername ist Deine Domain, das Passwort das STRATO Masterpasswort. Das legst Du im STRATO Backend unter **Sicherheit > Passwörter festlegen > Masterpasswort** fest.



So stellst Du SFTP in Filezilla ein

Auch wenn es etwas Mühe macht: Erlaube Browsern und FTP-Clients niemals, Passwörter auf der Festplatte zu speichern. Gib sie stattdessen bei der jeweiligen Verbindung immer neu ein. Denn zu leicht könnte diese Passwort-Datei in falsche Hände geraten.

## Veraltete Dateien am Webserver

Besonders beliebt bei Hackern sind veraltete und vergessene Dateien auf Webservern. Eventuell hast Du früher einmal irgendeine App ausprobiert oder ein zusätzliches WordPress zum Testen installiert, das Du längst nicht mehr nutzt. Auch wenn diese Dateien von keiner Webseite her verlinkt sind: Hacker kennen die Standardpfade und Dateinamen von alten App-Versionen mit Sicherheitslücken und finden solche Dateien über automatisierte Such-Scripts.

Kümmere Dich deshalb um Datenhygiene in Deinem Webspace: Lösche alles, was Du nicht mehr benutzt und halte alles andere mit Updates ständig auf dem aktuellsten Stand.

## 5.2 Potenzielle Sicherheitslücken von WordPress schließen

Um WordPress sicherer zu machen, kannst Du Login-Fehlermeldungen deaktivieren, einen Verzeichnisschutz für den Login-Bereich erstellen und das Tabellenpräfix ändern.

### Login-Fehlermeldungen deaktivieren

Wenn Du Dich mit falschen Benutzernamen anmeldest, gibt WordPress folgende Fehlermeldung aus: »FEHLER: Ungültiger Benutzername.« Und wenn Du den richtigen Benutzernamen und das falsche Passwort eingibst, lautet die Antwort: »FEHLER: Das Passwort, das du für den Benutzernamen XXX eingegeben hast, ist nicht korrekt.« Diese Informationen sind für Angreifer wertvoll und sollten deshalb deaktiviert werden. Um stattdessen zum Beispiel ein »Nachts sind alle Katzen grau.« einzublenden, fügst Du einfach folgenden Code ans Ende der **functions.php** Deines Themes ein:

```
function no_wordpress_errors() {  
    return ‚Nachts sind alle Katzen grau.‘;  
}  
add_filter( ‚login_errors‘, ‚no_wordpress_errors‘ );
```

### Verzeichnisschutz für Login-Bereich erstellen

Der sensible Login-Bereich wird standardmäßig über **/wp-login.php** aufgerufen. Das wissen sich Hacker und Botnetze zunutze zu machen. Mit einem Verzeichnisschutz per **.htaccess** baust Du ein Schloss davor, das unabhängig von WordPress funktioniert. Bei einem Brute-Force-Angriff wird so die Performance Deines Blogs nicht beeinträchtigt, weil der Angreifer es gar nicht erst bis zum Login-Bereich schafft.

Dazu erstellst Du in einem Code-Editor eine leere Datei, nennst diese **.htpasswd** und öffnest den [Htpasswd Generator](#). Dort gibst Du einen beliebigen Nutzernamen und ein Passwort ein. Das Tool verschlüsselt das Passwort und gibt eine Kombination aus Nutzernamen und Datensalat aus. Diese kopierst Du in die **.htpasswd**-Datei und lädst sie in das root-Verzeichnis Deines Webspace.

Im nächsten Schritt öffnest Du die **.htaccess** und fügst ans Ende folgenden Code ein:

```
<Files wp-login.php>
AuthType Basic
AuthName „Verzeichnisschutz“
AuthUserFile ABSOLUTERPFAD/.htpasswd
Require valid-user
</Files>
```

Bei **AuthUserFile** musst Du statt ABSOLUTERPFAD Deinen individuellen absoluten Pfad zur **.htpasswd** angeben (`www.wunschdomain.de/.htpasswd` funktioniert nicht). Was das ist und wie Du den herausfindest, erklären wir in [dieser FAQ](#). Nach dem Upload der bearbeiteten **.htaccess** erwartest Dich beim Login-Versuch folgende Meldung:

Der Verzeichnisschutz funktioniert unabhängig von WordPress.

Damit ist Dein Login-Bereich zusätzlich geschützt. Um Dich einloggen zu können, musst Du den oben erstellten Benutzernamen und das (unverschlüsselte) Passwort eingeben. Falls Du die Zugangsdaten vergessen solltest, kannst Du die **.htpasswd**-Datei einfach durch eine neue ersetzen.

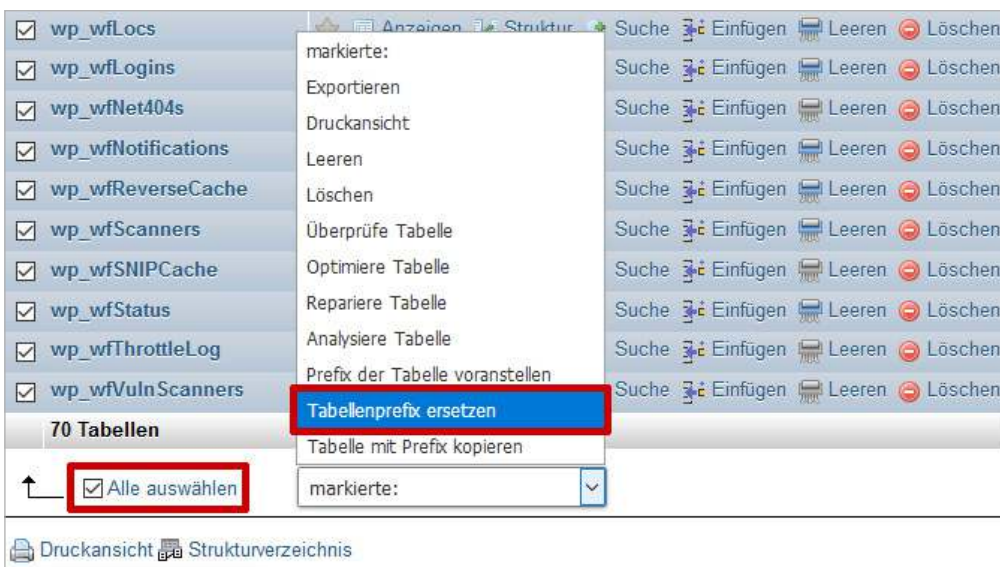
## Tabellen-Präfix ändern

WordPress vergibt bei der Installation für die Datenbank-Tabellen standardmäßig das Präfix **wp\_**. Diese potenzielle Sicherheitslücke können Dritte in Form von SQL-Injections ausnutzen. Bei einer manuellen Installation sollte deshalb am besten gleich ein anderes Präfix zum Einsatz kommen ([siehe Kapitel 1.1](#)).

Vorher unbedingt ein [Backup der Datenbank](#) machen, da bei einem Fehler die Website nicht mehr erreichbar ist!

So änderst Du das Tabellen-Präfix einer schon bestehenden WordPress-Installation:

1. Als Erstes benennst Du wie beschrieben das Präfix in der **wp-config.php** um. In unserem Beispiel wählen wir als Präfix **vogon\_**, sodass die Zeile wie folgt aussieht:  
**\$table\_prefix = ‚vogon\_‘;**
2. Danach rufst Du über den STRATO Kunden-Login phpMyAdmin auf: **Paket wählen > Datenbanken und Webspaces > Datenbankverwaltung > verwalten** (rechts neben der entsprechenden Datenbank > **Struktur**). Setze nun bei allen Tabellen ein Häkchen und wähle unten im Dropdown-Menü den Eintrag **Tabellenprefix ersetzen** ersetzen. Danach kannst Du das alte gegen das neue Präfix (hier: **vogon\_**) ersetzen.



Datenbankverwaltung phpMyAdmin

3. Im Reiter SQL gibst Du anschließend die folgenden beiden Befehle ein (statt **vogon\_** Dein neues Präfix):

```
UPDATE vogon_options SET option_name =
REPLACE(option_name, ‚wp_‘, ‚vogon_‘);
UPDATE vogon_usermeta SET meta_key =
REPLACE(meta_key, ‚wp_‘, ‚vogon_‘);
```

## Missverständnis: »Viel hilft viel«

Bei allen Risiken solltest Du dennoch nicht dem Missverständnis aufsitzen, dass viel automatisch auch viel bringt. Absolute Sicherheit gibt es nicht und jede Sicherheitsmaßnahme hat ihren Preis – ob Zeit und Nerven, Geld oder Performance-Einbußen durch Sicherheitsmaßnahmen. Zwei oder drei parallel installierte Sicherheits-Plugins beispielsweise machen Dein Blog unverhältnismäßig langsam und kosten damit Besucher und Google-Ranking.

Sicherheit ist immer eine Risikoabwägung. Der Aufwand muss verhältnismäßig bleiben. Für einen Webshop mit hohen Umsätzen kann schon ein Ausfall von einer Stunde ein Desaster sein, während ein Hobby-Blogger hackerbedingt auch mal eine Woche Offline-Zeit verschmerzen kann.

Wäge also bei Sicherheitsmaßnahmen das Risiko in Deiner persönlichen Situation ab: Was passiert konkret, wenn ich gehackt werde? Und wie hoch ist der Wiederherstellungsaufwand? Für ein Hobby-Blog reichen vielleicht schon regelmäßige Backups aus, sodass sich das Blog nach einem Hackerangriff wiederherstellen lässt. Eine Shop-Website, die sensible Kundendaten verarbeitet, muss unbedingt verhindern, dass diese Daten in falsche Hände geraten.

## Zu viel vermeintliche Sicherheit kann auch schaden

Viele Sicherheits-Tutorials für WordPress, die man im Internet findet, berücksichtigen diese Risikoabwägung nicht. Die dort beschriebenen, umfangreichen Maßnahmen verursachen in den meisten Fällen mehr Probleme, als sie potenziell der Sicherheit zuträglich sind.

Da wird beispielsweise empfohlen, das Plugin-Verzeichnis umzubenennen oder die Datei **wp-config.php** am Webserver in ein höheres, aus dem Web nicht zugängliches Verzeichnis zu verlegen. Weiß man hier nicht ganz genau, was man tut und versteht die technischen Zusammenhänge nicht hundertprozentig, ist das Risiko größer als der Nutzen. Beispielsweise funktionieren manche Plugins plötzlich nicht mehr oder WordPress lässt sich überhaupt nicht mehr aufrufen.

Bei allem Streben nach Sicherheit: Das primäre Ziel ist der Spaß am Bloggen, nicht die Abwehr von Hackern. Gesundes Augenmaß ist daher wichtiger.

## 5.3 Typische WordPress-Fehler beheben

WordPress ist eine ausgereifte Software, die bei richtiger Anwendung kaum Probleme macht. Fehler durch Updates, Plugins, Themes oder Nutzer können dennoch dazu führen, dass WordPress eingeschränkt oder gar nicht mehr funktioniert. Wir zeigen Dir, wie Du die häufigsten Bugs behebst.

### White Screen of Death

Der sogenannte White Screen of Death ist ein Fehler, mit dem viele WordPress-Blogger irgendwann konfrontiert werden. Wie der Name vermuten lässt, bleiben dabei das Front- und/oder das Backend beim Aufruf weiß. Das heißt, dass Du WordPress nicht mehr nutzen kannst und auch Besucher nichts zu sehen bekommen. Verursacht wird das Problem häufig durch einen PHP-Fehler in einem Theme oder Plugin.

#### Ursache finden: Debug-Modus aktivieren

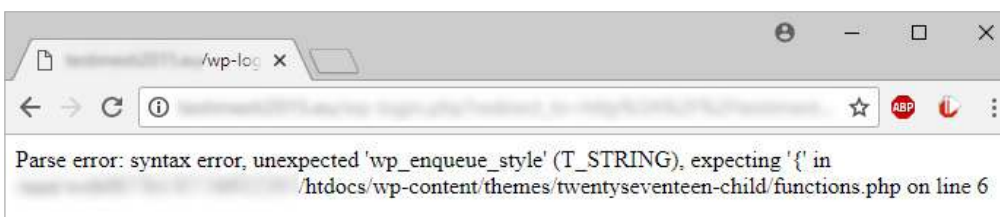
Um Informationen über die Ursache zu erhalten, solltest Du als Erstes den Debug-Modus aktivieren. Ändere dazu in der `wp-config.php` die Zeile

```
define('WP_DEBUG', false);
```

in

```
define('WP_DEBUG', true);
```

Wenn Du die Website nun erneut aufrufst, wird Dir eine detaillierte Meldung über den Fehler, die betroffene Datei und die Codezeile angezeigt.



In diesem Beispiel handelt es sich um einen Syntax-Error in der `functions.php` des Child-Themes.

### Lösung: Datei oder Verzeichnis umbenennen

Falls Du Zugriff auf das Backend hast, solltest Du das betroffene Theme oder Plugin deaktivieren. Andernfalls kannst Du die in der Fehlermeldung benannte Datei im Webservice umbenennen (z. B. mit dem Anhang .bak), damit WordPress die fehlerhafte Datei nicht mehr aufruft. Deine Einstellungen bleiben dabei aber erhalten.

Wenn Dein Theme betroffen ist, solltest Du das gesamte Theme-Verzeichnis umbenennen. WordPress greift dann automatisch auf ein anderes Theme zurück.

Nicht nur aus diesem Grund muss immer mindestens ein weiteres Theme vorhanden sein.

Vergiss nicht den Debug-Modus anschließend wieder auszuschalten, indem Du den Wert in der oben genannten Zeile wieder zurück auf false setzt. Deine Besucher bekommen sonst unschöne (Fehler)Meldungen zu sehen, die für Entwickler gedacht sind.

Tipp: Source-Code-Editoren können Dich bei der Fehleranalyse unterstützen oder sorgen dafür, fehlerhaften Code bereits bei der Eingabe aufzuzeigen. Wie Du gute Editoren findest, liest Du [hier](#).

### »Dem Theme fehlt das Stylesheet style.css«

Du willst ein neues Theme installieren und bekommst die Meldung, dass das Stylesheet fehlt? Mit dem folgenden Vorgehen funktioniert die Installation.

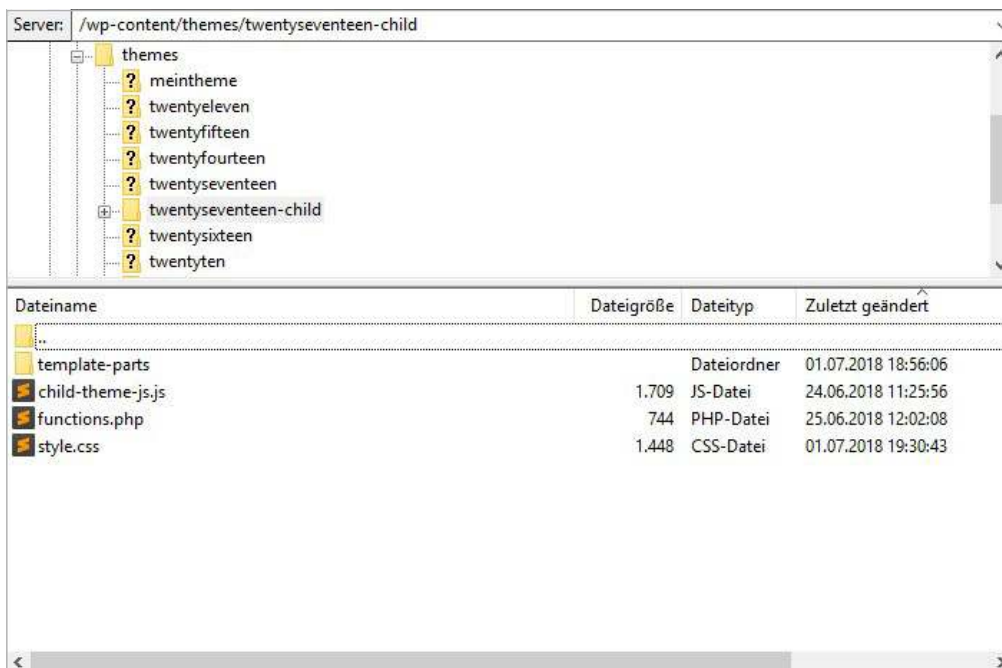
Die Fehlermeldung ist selbsterklärend: »Das Paket konnte nicht installiert werden. Dem Theme fehlt das Stylesheet style.css.« Folglich kannst Du das neue Theme nicht nutzen.

### Ursache: Fehlende oder fehlerhafte style.css

Das heißt, dass WordPress die Datei **style.css** nicht finden kann oder dass die Datei fehlerhaft ist. Bei einem Theme aus der [WordPress Repository](#) ist beides unwahrscheinlich. Vielleicht hast Du das (Child-)Theme ja [selbst entwickelt](#)?

## Lösung: Korrekte style.css bereitstellen

Stelle sicher, dass die **style.css** im Hauptverzeichnis Deines (Child-)Themes liegt. Die **style.css** und die **functions.php** sollten in der obersten Verzeichnisebene liegen.



WordPress speichert die Themes im gleichnamigen Unterverzeichnis. Wie in diesem Beispiel zu sehen ist, liegen **functions.php** und **style.css** im Hauptverzeichnis des Child-Themes.

Übrigens: Wie die **style.css** aussehen muss, haben wir [hier beschrieben](#). Das Wichtigste ist die Zeile Template: Hier muss der exakte Name des Verzeichnisses angegeben sein, in dem das ausgewählte Theme liegt. Für das Theme Twenty Seventeen etwa wäre das **twentyseventeen**.

## 404-Fehler

Sogenannte 404-Fehlerseiten treten häufig auf. Das Problem ist vergleichsweise harmlos und in WordPress schnell gefixt.

### Ursache: Seite existiert nicht

Ein 404-Error tritt auf, wenn die URL einer Seite oder eines Beitrags fehlerhaft ist oder die entsprechende Datei fehlt. Bei den meisten WordPress-Themes erscheint dann der Standard-Hinweis: »Die Seite konnte leider nicht gefunden werden.«

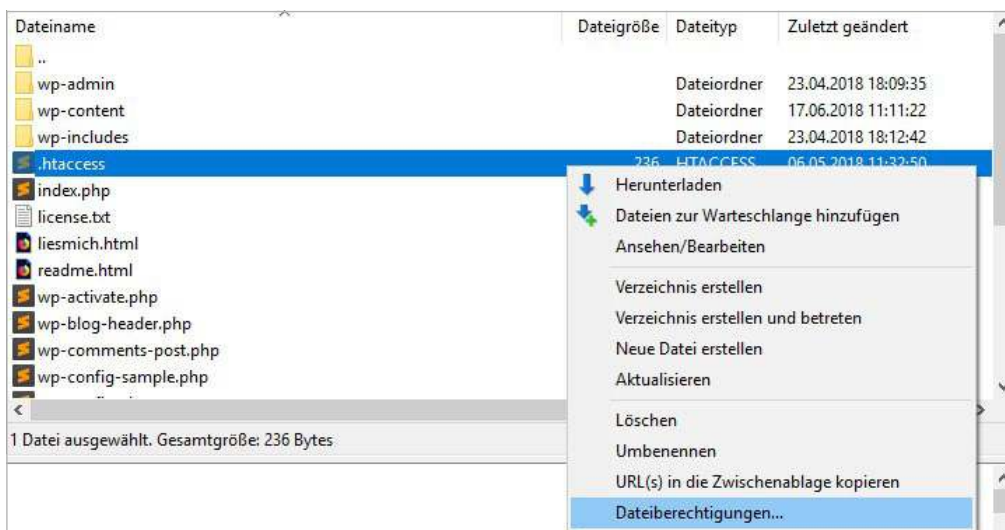


Mit ein paar PHP-Kenntnissen kannst Du diese Seiten sogar [individuell gestalten](#). Dennoch ist es aber natürlich besser, wenn Deine Besucher diese Seiten gar nicht erst zu sehen bekommen.

### Lösung 1: ULRs neu generieren

Klicke im Backend unter **Einstellungen > Permalinks** auf **Änderungen speichern**, damit WordPress sämtliche URLs Deiner Beiträge und Seiten neu generiert. In den meisten Fällen ist das Problem damit schon behoben.

Wenn das nicht hilft, sind vielleicht die Berechtigungen der **.htaccess** im Hauptverzeichnis Deiner WordPress-Installation zu restriktiv. Klicke im FTP-Programm mit der rechten Maustaste auf die Datei und wähle dann im Menü **Dateiberechtigungen**.



Gib nun im Fenster »Dateiattribute ändern« als numerischen Wert 666 ein und wiederhole das Vorgehen oben (**Einstellungen > Permalinks > Änderungen speichern**).

Setze die Dateiberechtigungen anschließend auf die ursprünglichen Einstellungen zurück!

## Lösung 2: Weiterleitungen erneuern

Ist das Problem damit immer noch nicht gelöst, ist wahrscheinlich die `.htaccess`-Datei fehlerhaft. Lade die Datei herunter und gib dort folgende Zeilen ein, damit WordPress die URLs korrekt generieren kann:

```
# BEGIN WordPress
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ./index.php [L]
</IfModule>
# END WordPress
```

Tipp: Wenn Du Dir nicht sicher bist, welche URL zu einer Fehlerseite führen, hilft Dir unser kostenloser [»Linkchecker](#). Auch mit dem Plugin [»Redirection](#) kommst Du fehlerhaften URLs schneller auf die Spur.

## »Fatal error undefined function is\_network\_admin()«

Wenn Du diese Fehlermeldung siehst, geht erst mal nichts mehr. Doch keine Sorge, mit den folgenden Schritten bringst Du Dein WordPress wieder zum Laufen.

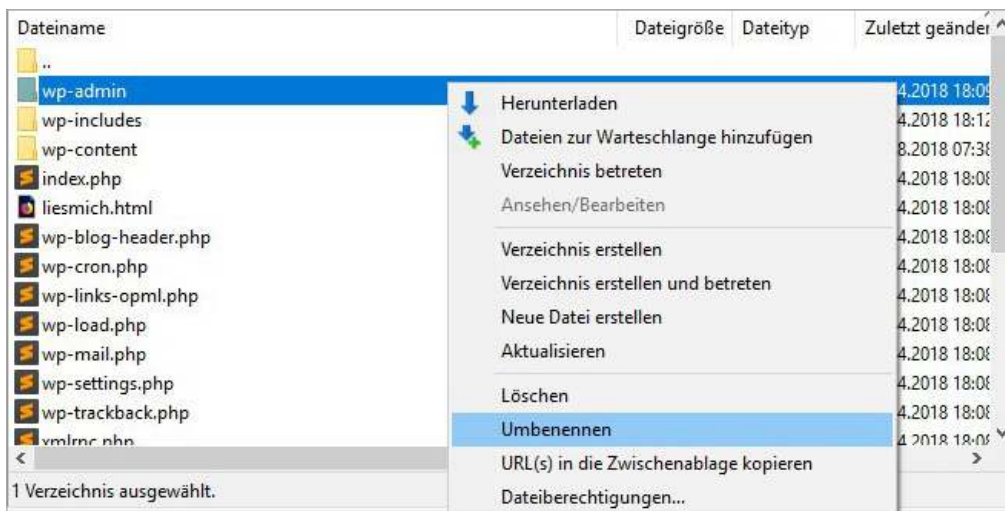
### Ursache: Fehlerhaftes Update

Die Fehlermeldung »Fatal error undefined function is\_network\_admin()« kann auftreten, wenn Du Dich nach einem automatischen WordPress-Update versuchst einzuloggen. Das Update ist in diesem Fall fehlgeschlagen und Deine Installation beschädigt.

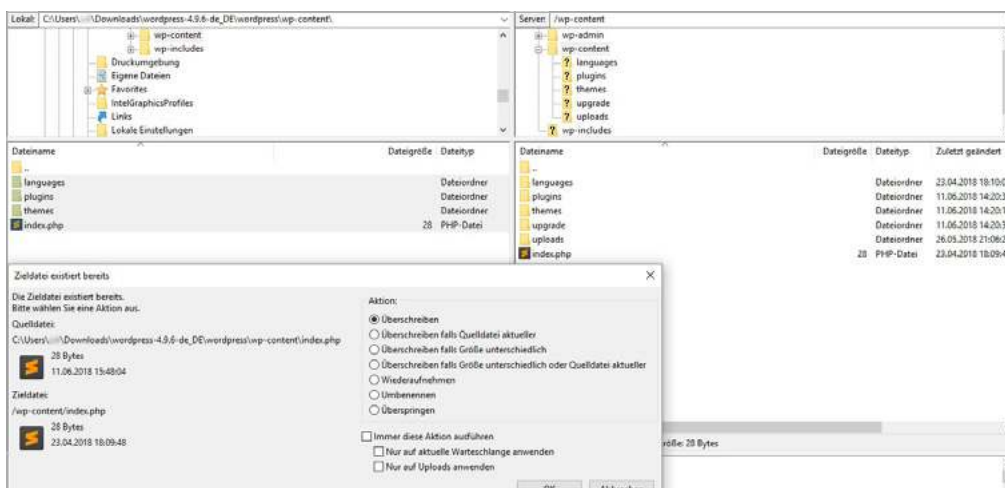
### Lösung: Dateien und Verzeichnisse austauschen

1. Erstelle als Erstes ein Backup Deiner WordPress-Installation. Wie das funktioniert, haben wir [»hier beschrieben](#).
2. Lade WordPress [»in der aktuellen Version](#) herunter und entpacke die `.zip`-Datei.

- Benenne per FTP die Ordner **wp-admin** und **wp-includes** im Hauptverzeichnis Deiner WordPress-Installation um (z. B. mit dem Anhang .bak). Klicke dazu mit der rechten Maustaste auf das entsprechende Verzeichnis und wähle dann **Umbenennen**.



- Lösche anschließend die Ordner **wp-admin** und **wp-includes** aus dem Hauptverzeichnis und lade diese beiden Ordner aus dem gerade heruntergeladenen WordPress hoch.
- Lade dann das Verzeichnis **wp-content** hoch und ersetze sämtliche alten Dateien durch die neuen. Deine Child-Themes und Plugins bleiben dabei erhalten. Lösche darüber hinaus keine weiteren Dateien oder Ordner.



6. Lade dann noch alle Dateien auf der Ebene des Hauptverzeichnisses hoch. Achte darauf, nicht noch einmal Verzeichnisse mit hoch zu laden. Keine Sorge: Deine **wp-config.php** wird dabei nicht verändert.
7. Prüfe, ob in der **wp-config-sample.php** neue Einstellungen existieren, die Du Deiner **wp-config.php** hinzufügen möchtest.
8. Lösche die Datei **.maintenance**.
9. Logge Dich in Dein WordPress ein und folge den weiteren Anweisungen.

Wenn Du ein Caching-Plugin verwendest, musst Du abschließend den Cache leeren.

### »Error establishing a database connection«

WordPress benötigt für den Betrieb eine **mySQL**-Datenbank. Darin werden Deine Texte gespeichert und beim Aufrufen sofort angezeigt. Kann die Verbindung nicht hergestellt werden, siehst Du stattdessen diese Fehlermeldung:



#### Ursachen: Falsche Zugangsdaten, Datenbank defekt/gelöscht/nicht erreichbar

Es kommen verschiedene Ursachen in Frage, warum WordPress keine Verbindung zur Datenbank aufbaut: Die Zugangsdaten können falsch sein, vielleicht ist die Datenbank defekt oder der Datenbank-Server nicht erreichbar. Möglich ist auch, dass die Datenbank gelöscht wurde.

#### Lösung 1: **wp-config.php** korrigieren

Alle Informationen zur Datenbank findest Du im [STRATO Kunden-Login](#) unter **Paket > Datenbanken und Webspaces > Datenbankverwaltung**. Prüfe, ob Datenbankname und Benutzername mit den Einträgen in der Datei **config.php** übereinstimmen. Diese Datei liegt im Hauptverzeichnis Deiner WordPress-

Installation. Als Serveradresse muss **rdbms.strato.de** (statt **localhost**) eingetragen sein. Wenn alle Einträge korrekt sind, ändere das Passwort und trage das neue Passwort in die **config.php** ein.

```
// ** MySQL-Einstellungen ** //  
/** Diese Zugangsdaten bekommst du von deinem Webhoster. **/  
  
/**  
 * Ersetze datenbankname_hier_einfuegen  
 * mit dem Namen der Datenbank, die du verwenden möchtest.  
 */  
define('DB_NAME', 'XXXXXXXX');  
  
/**  
 * Ersetze benutzername_hier_einfuegen  
 * mit deinem MySQL-Datenbank-Benutzernamen.  
 */  
define('DB_USER', 'XXXXXXXX');  
  
/**  
 * Ersetze password_hier_einfuegen mit deinem MySQL-Passwort.  
 */  
define('DB_PASSWORD', 'XXXXXXXX');  
  
/**  
 * Ersetze localhost mit der MySQL-Serveradresse.  
 */  
define('DB_HOST', 'rdbms.strato.de');
```

MySQL-Einträge in der wp-config.php

In der Regel sollte das Problem damit gelöst sein. Andernfalls ist der Datenbank-Server womöglich temporär nicht erreichbar. Das kann zum Beispiel passieren, wenn Deine Website sehr viele Besucher hat und der Server überlastet ist. Mit einem [Webanalyse-Programm](#) wie Google Analytics erfährst Du in Echtzeit, wie viele Besucher Deine Website hat.

## Lösung 2: Datenbank-Backup wiederherstellen

Wenn der bisher beschriebene Weg nicht funktioniert, kannst Du ein (hoffentlich vorhandenes) Datenbank-Backup wiederherstellen. Wie Du Backups anlegst, haben wir [hier beschrieben](#).

In der Zwischenzeit hinzugefügte oder veränderte Inhalte und Kommentare von Besuchern Deines Blogs gehen bei der Wiederherstellung unwiederbringlich verloren!

Wenn Du WordPress mit der Installationshilfe AppWizard installiert hast, kannst Du das Tool **STRATO BackupControl** nutzen. Wähle dazu im Kunden-Login **Sicherheit > STRATO BackupControl > Anwendungen (AppWizard)**. Nachdem Du Deine WordPress-Installation im Dropdown-Menü ausgewählt hast, werden Dir bis zu zwei Sicherungskopien angezeigt. Mit einem Klick auf **zurückspielen** versetzt das Tool Datenbank und Webspaces auf den entsprechenden Stand zurück.

### Backups

**i** Für Ihre Installation stehen Ihnen 2 Backup Speicherplätze zur Verfügung. Je einer für ein automatisches und ein manuelles Backup. Bitte beachten Sie, dass die Erstellung eines neuen Backups jeweils das bestehende Backup in dem entsprechenden Speicherplatz überschreibt.

Automatische Backups erstellen

- wöchentlich
- monatlich

**Speichern**

Datum	Bezeichnung	Erstellung	Version	
22.04.2018 16:58	WordPress_SecureMode	Automatisch durch Aktualisierung	4.9.5	<b>zurückspielen</b>

## 5.4 Statische Websites mit WordPress erstellen

Statische Websites bieten weniger Angriffsfläche und laden schneller. Wir zeigen Dir, wie Du Deine WordPress-Seiten in wenigen Schritten konvertierst.

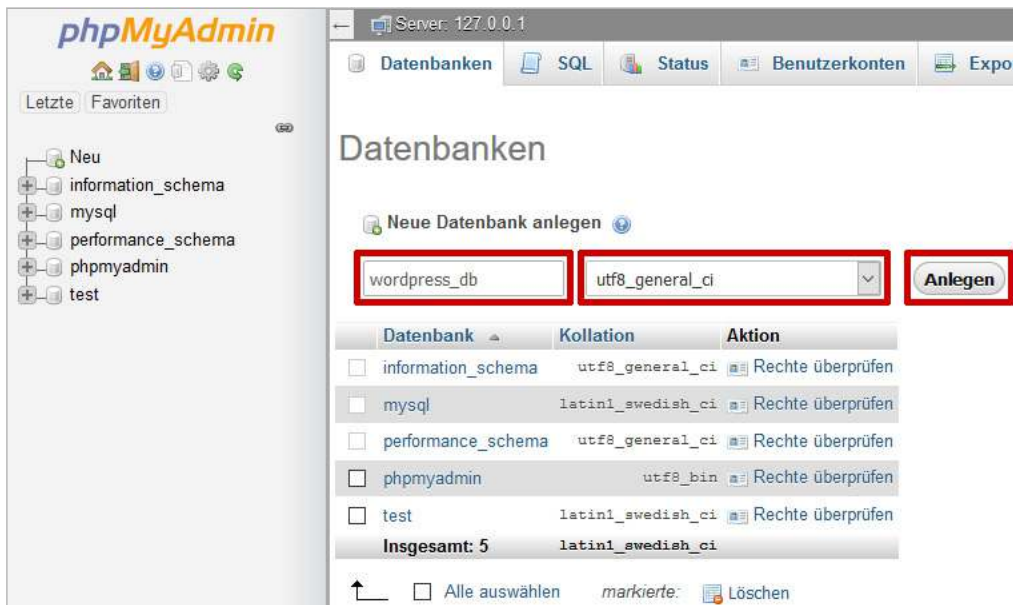
Als eines der populärsten Content-Management-Systeme ist WordPress ein beliebtes Angriffsziel. Viele Schwachstellen sind bekannt, andere kommen mit neuen Updates. Wer auf Nummer sichergehen will, installiert das System lokal, exportiert statische Seiten und lädt diese auf den Server. Das lohnt sich vor allem für Websites, deren Inhalte sich selten ändern. Willkommener Nebeneffekt: Ohne PHP und Datenbank sind die Ladezeiten meist deutlich kürzer.

Auf dynamische Funktionen (z. B. Kommentare) musst Du dann allerdings verzichten. Für ein Blog ist das folgende Vorgehen daher weniger geeignet.

### Schritt 1: XAMPP einrichten

Zunächst installierst Du WordPress lokal auf Deinem Computer. Um Server und Datenbank zu emulieren verwendest Du am besten [Xampp](#). Nach der Installation startest Du die beiden Module Apache und MySQL. Das Programm rufst Du anschließend über den Browser auf und gibst dazu in die Browserzeile »localhost« ein.

Die Datenbankverwaltung erreichst Du über den Menüpunkt **phpMyAdmin** oben rechts. Über den Reiter **Datenbanken** kannst Du nun eine neue Datenbank anlegen. Die Datenbank bekommt einen beliebigen Namen (unbedingt notieren!), bei der Option **Kollation** sollte **utf8\_general\_ci** gewählt werden. Nach einem Klick auf **Anlegen** erscheint die frische Datenbank auf der linken Seite.



phpMyAdmin: Datenbank anlegen leicht gemacht

Nun musst Du noch einen Benutzer anlegen. In der Menüleiste oben wählst Du den Eintrag **Rechte** und dann den Link **Benutzerkonto hinzufügen**. Gib einen Namen ein, wähle im Dropdown-Menü als Host Lokal und dann ein sicheres Passwort (ebenfalls beides notieren!). Unter **Datenbank für Benutzer** muss bei **Gewähre alle Rechte auf die Datenbank** ein Häkchen gesetzt werden. Abschließend bestätigst Du das Ganze mit **OK**.



## Benutzerkonto hinzufügen

**Anmeldeinformation**

Benutzername: Textfeld verwenden:

Hostname: Lokal  ⓘ

Passwort: Textfeld verwenden:  Strength: ■■■■■ Stark

Wiederholen:

Authentifizierungs Plugin: Native MySQL-Authentifizierung

Passwort generieren: Generieren

**Datenbank für Benutzerkonto**

Erstelle eine Datenbank mit gleichem Namen und gewähre alle Rechte.

Gewähre alle Rechte auf Datenbanken die mit dem Benutzernamen beginnen (username\\_%).

Gewähre alle Rechte auf die Datenbank wordpress\_db.

## Schritt 2: WordPress installieren

Nun kannst Du WordPress installieren. Dazu lädst Du die [aktuelle Version](#) herunter, entpackst und kopierst den WordPress-Ordner in das htdocs-Verzeichnis von Xampp. Wo sich das befindet, hängt ab vom Ort Deiner Xampp-Installation (z. B. C:\xampp\htdocs). Benenne das Verzeichnis in »wordpress« um. Wenn Du nun **localhost/wordpress/wp-admin/setup-config.php** aufrufst, musst Du nur noch die (hoffentlich) notierten Datenbank- und WordPress-Login-Informationen eingeben. Deine lokale WordPress-Installation erreichst Du (nach dem Start der Apache- und MySQL-Module) anschließend immer unter <http://localhost/wordpress/>.

## Schritt 3: Simply Static

Im letzten Schritt installierst Du ein Plugin, mit dem Du WordPress-Seiten in statische umwandelst. Das geht zum Beispiel mit [Simply Static](#). Der Name ist hier Programm: Installieren, aktivieren und unter **Simply Static > Settings > Destination-URL** als absolute URL die Domain eingeben, unter welcher die statischen Seiten

erreichbar sein sollen. Das Plugin schreibt dann die URLs der Dateien und Links von localhost/wordpress entsprechend um.

Simply Static > Settings

General | Include/Exclude | Advanced | Reset

**Destination URLs**

When exporting your static site, any links to your WordPress site will be replaced by one of the following: absolute URLs, relative URLs, or URLs constructed for offline use.

Use absolute URLs

Convert all URLs for your WordPress site to absolute URLs at the domain specified above.

Use relative URLs

Convert all URLs for your WordPress site to relative URLs that will work at any domain. Optionally specify a path above if you intend to place the files in a subdirectory.  
Example: enter `/path/` above if you wanted to serve your files at `www.example.com/path/`.

Save for offline use

Convert all URLs for your WordPress site so that you can browse the site locally on your own computer without hosting it on a web server.

**Delivery Method**

Saving your static files to a ZIP archive is Simply Static's default delivery method. After generating your static files you will be provided with a link to download the ZIP archive.

[Save Changes](#)

Simply Static: Neben einer absoluten Adresse kannst Du auch relative URLs angeben. Das ist zum Beispiel sinnvoll, wenn Du die Domain noch nicht kennst oder die Inhalte für verschiedene Domains verwenden willst.

Danach kannst Du über Generate mit dem Export loslegen. Vorsicht: Die Seiten werden in einer verschachtelten Verzeichnisstruktur gespeichert. Beim FTP-Upload solltest Du nur den Inhalt des Ordners mit der index.html (samt Unterverzeichnissen) übertragen.

## Lokales WordPress als HTML-Generator und Test-System

Mit der hier beschriebenen Vorgehensweise sind nur die exportierten statischen Seiten öffentlich, die WordPress-Installation bleibt für Besucher und potenzielle Angreifer unsichtbar. Allerdings musst Du neue Beiträge und Änderungen immer manuell hochladen oder diesen Upload mit einem lokalen Skript oder einer Synchronisierungssoftware auf Deinem Computer automatisieren. Dynamische Elemente wie Kontaktformular und Kommentarfunktion gibt es nicht.

WordPress lokal zu nutzen, ist auch zu Testzwecken empfehlenswert. Es ist sicherer und professioneller, Änderungen am Blog immer erst zu testen. Wenn Du also bereits einen Blog hast, folgst Du am besten gleich den ersten beiden Schritten und erstellst eine lokale Kopie. Der Aufwand lohnt sich.

## 5.5 Plugins als Sicherheitsrisiko

Leider sind Plugins für WordPress nicht nur nützlich bis unentbehrlich – sie sind auch das größte Sicherheitsrisiko. Denn es gibt nahezu keine Qualitätskontrolle der angebotenen Plugins und die Qualifikationen der Programmierer in Hinblick auf Sicherheit variieren stark, ohne dass Du das als Anwender erkennen kannst.

Deshalb solltest Du sowohl aus Performance- als auch Sicherheitsgründen so wenig Plugins wie möglich verwenden und auf die Funktionalitäten reduzieren, die wirklich unverzichtbar sind.

Allgemeine Tipps, um das Plugin-Risiko zu minimieren:

- Prüfe neue Plugins möglichst genau. Positive Indizien sind hohe Nutzerzahlen, gute Bewertungen, wenig ernsthafte Probleme in den Support-Foren und regelmäßige Aktualisierungen des Plugins.
- Je umfassender die Funktionalität eines Plugins ist, desto wahrscheinlicher schleichen sich auch bei guten Programmierern Sicherheitslücken ein. Vermeide deshalb Plugins mit zahlreichen Funktionen, die Du größtenteils ohnehin nicht nutzt.
- Installiere generell keine Plugins, die dem User irgendwelche Datei- oder Bilder-Uploads erlauben.
- Lösche unbenutzte Plugins. Deaktivieren genügt nicht.

# 6 Performance

## 6.1 WordPress Beine machen per .htaccess

Plugins, Themes, Code Snippets – all das erleichtert den Blogger-Alltag, kann WordPress aber langsam machen. Mit der Folge, dass Besucher vorzeitig abspringen und Suchmaschinen Deine Website mit schlechteren Positionen abstrafen. Im Folgenden zeigen wir Dir, wie Du die Performance von WordPress über die **.htaccess**-Datei verbesserst.

### 1. .htaccess-Datei herunterladen

Die **.htaccess**-Datei ist eine Konfigurationsdatei für Webserver. Damit kannst Du unter anderem das Browser-Caching konfigurieren und die Komprimierung aktivieren. WordPress erstellt die **.htaccess** bei der Installation automatisch und legt sie im Hauptverzeichnis ab. Lade die Datei mit Deinem FTP-Programm herunter und öffne sie mit einem Text-Editor.

Speichere die Original-Version, um im Notfall den alten Zustand wiederherstellen zu können.

### 2. Browser-Caching nutzen

Beim Browser-Caching legt der Browser Kopien von CSS-, JavaScript- und Grafikdateien im lokalen Speicher ab. Wenn ein Besucher die Seite erneut aufruft, muss der Browser diese nicht erneut vom Server herunterladen. Das spart Traffic und Ladezeit, heißt aber auch: Der Browser eines wiederkehrenden Besuchers aktualisiert zwischenzeitig geänderte Dateien erst nach Ablauf der eingestellten Frist. In den meisten Fällen sollte das aber kein Problem darstellen, sodass Du großzügige Werte (z. B. einen Monat) verwenden kannst.

Das Browser-Caching konfigurierst Du in der **.htaccess**-Datei. In der Regel stehen dort folgende Zeilen:

```
# BEGIN WordPress
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ./index.php [L]
</IfModule>
# END WordPress
```

Eigenen Code kannst Du einfach darunter eintragen. Mit der Anweisung **ExpiresByType** gibst Du für die verschiedenen Dateitypen an, wie lange die lokalen Kopien gespeichert werden. Im nachfolgenden Code-Beispiel beträgt die Dauer jeweils einen Monat.

```
# Browser Caching
<IfModule mod_expires.c>
ExpiresActive On
ExpiresByType image/gif „access plus 1 months“
ExpiresByType image/ico „access plus 1 months“
ExpiresByType image/jpeg „access plus 1 months“
ExpiresByType image/jpg „access plus 1 months“
ExpiresByType image/png „access plus 1 months“
ExpiresByType text/css „access plus 1 months“
ExpiresByType text/javascript „access plus 1 months“
ExpiresByType application/x-javascript „access plus 1 months“
ExpiresByType application/javascript „access plus 1 months“
</IfModule>
```

Nach dem Upload der **.htaccess** kannst Du mit [Google Page Speed Insights](#) prüfen, ob Deine Anweisungen funktionieren. Das Tool schlägt in dem Fall nicht mehr vor, Browser-Caching zu nutzen.

### 3. Komprimierung aktivieren

Die Komprimierung ist ebenfalls unkompliziert umzusetzen. Zur Methode: Das Programm Gzip komprimiert HTML- und CSS-Dateien. Dadurch ist die von einer Website zu übertragene Datenmenge kleiner. Sobald der Browser die Dateien

empfangen hat, entpackt er sie wieder. Weil das Zeit spart, solltest Du die Technik für Deine Website nutzen. Füge dazu einfach die folgenden Zeilen in die **.htaccess** ein:

```
# Gzip Compression
<IfModule mod_deflate.c>
  AddOutputFilterByType DEFLATE application/javascript
  AddOutputFilterByType DEFLATE application/rss+xml
  AddOutputFilterByType DEFLATE application/vnd.ms-fontobject
  AddOutputFilterByType DEFLATE application/x-font
  AddOutputFilterByType DEFLATE application/x-font-opentype
  AddOutputFilterByType DEFLATE application/x-font-otf
  AddOutputFilterByType DEFLATE application/x-font-truetype
  AddOutputFilterByType DEFLATE application/x-font-ttf
  AddOutputFilterByType DEFLATE application/x-javascript
  AddOutputFilterByType DEFLATE application/xhtml+xml
  AddOutputFilterByType DEFLATE application/xml
  AddOutputFilterByType DEFLATE font/opentype
  AddOutputFilterByType DEFLATE font/otf
  AddOutputFilterByType DEFLATE font/ttf
  AddOutputFilterByType DEFLATE image/svg+xml
  AddOutputFilterByType DEFLATE image/x-icon
  AddOutputFilterByType DEFLATE text/css
  AddOutputFilterByType DEFLATE text/html
  AddOutputFilterByType DEFLATE text/shtml
  AddOutputFilterByType DEFLATE text/javascript
  AddOutputFilterByType DEFLATE text/plain
  AddOutputFilterByType DEFLATE text/xml
</IfModule>
```

Nach dem Hochladen kannst Du noch mal Page Speed Insights nutzen, um zu prüfen, ob der Server die angegebenen Dateitypen tatsächlich komprimiert.

## Einfach schneller

Das Warten hat ein Ende. Mit den grundlegenden Tipps in unserem E-Book [WordPress für Einsteiger](#) (Kapitel 5.4) und den hier gezeigten Methoden sind kurze Ladezeiten garantiert. Und Deine mobilen Besucher dürften sich neben der Geschwindigkeit auch über das gesparte Datenvolumen freuen.

# 7 Tracking

## 7.1 An diesen Zahlen erkennst Du, ob Dein Blog erfolgreich ist

Zahlen und Statistiken verraten Dir, was auf Deinem Blog gut läuft und was nicht. Mit diesem Wissen kannst Du besser einschätzen, was Deine Leser wollen – aber auch, auf welchem Weg sie Dein Blog überhaupt finden und wie Du sie besser erreichen kannst. Wenn Du Kooperationen mit Firmen suchst oder Werbebanner akquirieren willst, fragen potenzielle Kunden meist nach einigen Erfolgskennzahlen, um einschätzen zu können, ob sich eine Zusammenarbeit lohnt.

Bei Statistiken kommt es wie immer aber darauf an, aus einem Berg von Informationen die wirklich relevanten Zahlen herauszufischen und die Werte richtig zu interpretieren.

Nicht absolute Zahlen, sondern Trends sind wichtig

Hohe, absolute Zahlen sind schön, zeigen aber nur den aktuellen Stand. Zum Messen Deines Erfolgs sind Trends und Entwicklungen viel interessanter. Sie eröffnen Dir nämlich den Blick in die Zukunft: Kommen neue Leser hinzu? Werden sie zu Stammlesern? Steigt die Zahl der Kommentare, Shares und Likes zu Deinen Beiträgen? Kurz: Bist Du auf dem richtigen Weg?



Die periodisch schwankenden Besucherzahlen in diesem Beispiel zeigen, dass an Wochenenden weniger los ist.

Trends und Entwicklungen sind aber auch deshalb interessanter, weil sie sich gut mit denen anderer Blogs vergleichen lassen. Der Analyse-Dienst [SimilarWeb](#) liefert Zahlen zu monatlichen Besucherzahlen Deiner und fremder Websites im Zeitverlauf – ideal für die Konkurrenzanalyse. Absolute Zahlen sind dagegen stark abhängig von Deinem Thema, eventuell auch von der Jahreszeit und anderen, individuellen Faktoren.

Dennoch brauchst Du als Grundlage zunächst einmal absolute Zahlen, die Du aus Services wie dem kostenlosen Google Analytics oder auch dem Open-Source-Tool Matomo beziehen kannst.

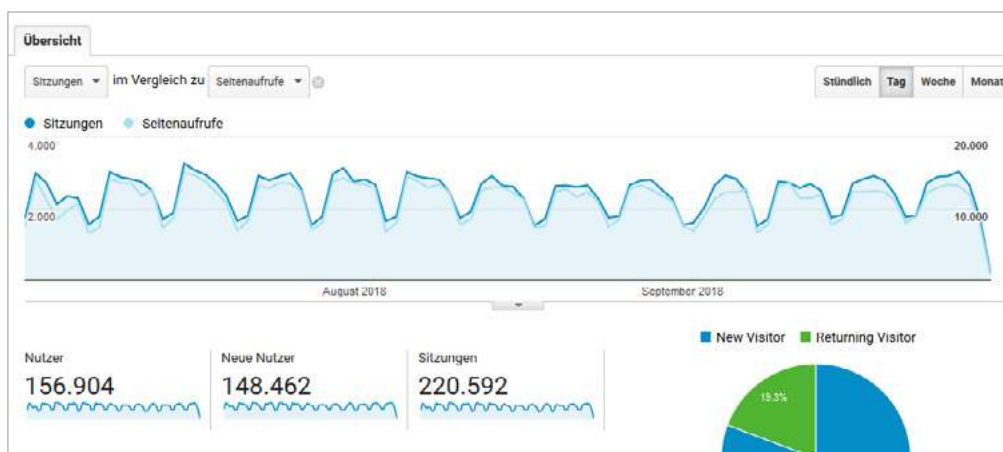
## 1. Visits und Unique Visits, Sitzungen und Nutzer

Für den Traffic gibt es drei grundlegende Kennzahlen, die üblicherweise monatlich zusammengefasst werden:

- Seitenaufrufe (Page Impressions): Hier zählt jede einzelne Seite, die Deine Besucher aufrufen.
- Sitzungen (Visits): Alle Seiten, die ein einzelner Besucher zusammenhängend aufruft, ergeben zusammen eine Sitzung.
- Nutzer (Unique Visitors) gibt an, wie viele unterschiedliche Menschen Dein Blog in einem ausgewählten Zeitraum besuchen.

Die Zahl der Seitenaufrufe für sich genommen ist ziemlich uninteressant, schon weil sich das leicht manipulieren lässt. Mehr dazu im Absatz »Aussagekraft statistischer Werte«.

Die viel härtere Währung ist die Zahl der Nutzer. Abgesehen von statistischen Ungenauigkeiten zeigt Dir diese Zahl, wie viele Menschen Du tatsächlich erreichst.



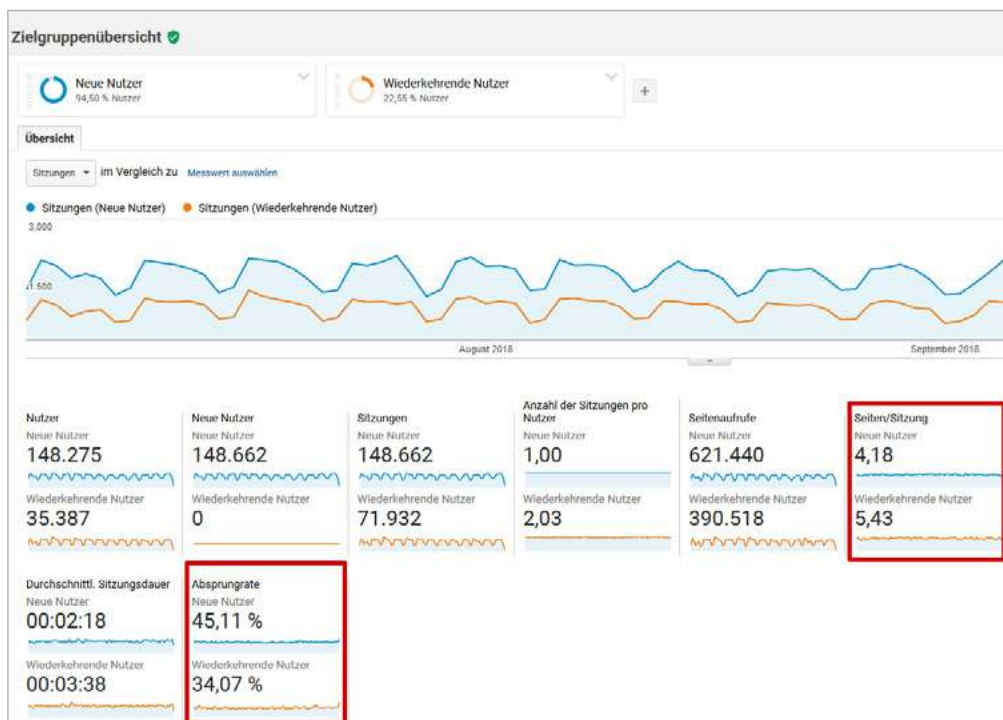
Übersicht der Traffic-Kennzahlen bei Google Analytics



## 2. Eintagsfliege oder Lesermagnet?

Eine bessere Aussage über die Qualität Deines Blogs liefert die Zahl der Seiten, die ein Besucher im Durchschnitt ansieht. Das sind übrigens überraschend wenig. Wenn Du einen Durchschnittswert von drei oder mehr erzielst, kannst Du stolz sein. Paradoxerweise ist dieser Wert tendenziell niedriger, wenn Du viele Stammleser hast. Denn Stammleser lesen nur den jeweils neuesten Beitrag, weil sie den Rest schon kennen.

Um herauszufinden, ob Dein Blog attraktiv für neue Leser ist, solltest Du die Analyse der Seiten pro Besuch auf neue Leser beschränken. In Google Analytics kannst Du das über **Zielgruppe > Übersicht** einstellen und auch die unterschiedlichen Werte miteinander vergleichen.



Seiten pro Sitzung und Absprungrate von Stammlesern und neuen Besuchern im Vergleich

Die **Absprungrate** (engl.: Bounce Rate) hängt eng damit zusammen. Diese Zahl zeigt Dir, wie viele Leser nur eine einzige Seite auf Deinem Blog ansehen und danach wieder gehen. Wie Du beim Thema Stammleser gesehen hast, ist das nicht unbedingt etwas Negatives und kann auch heißen, dass ein Leser, der über Google auf Deiner Seite gelandet ist, genau das gefunden hat, wonach er gesucht hat.

Einen weiteren Anhaltspunkt gibt die **durchschnittliche Sitzungsdauer**. Setzt Du die ins Verhältnis zur Zahl der Seiten pro Sitzung, kannst Du Dir ungefähr vorstellen, wie lange die Besucher sich mit einem einzelnen Beitrag in Deinem Blog beschäftigen. Nicht erschrecken: Auch hier sind die durchschnittlichen Werte recht niedrig – es sind eben Durchschnittswerte, die vor allem von Besuchern nach unten gedrückt werden, die irrtümlich auf Deinem Blog landen und deshalb schon nach zwei oder drei Sekunden wieder gehen.

Die Absprungrate ist aber auch ein Indiz dafür, wie gut es Dir gelingt, Leser zum Bleiben zu bewegen, beispielsweise durch weiterführende Links am Ende jedes Beitrags.

Wenn Du Dich tiefer in diesen Aspekt einarbeiten willst, dann analysierst Du die Absprungrate Deiner Beiträge einzeln. In Google Analytics findest Du eine entsprechende Liste unter **Verhalten > Website-Content > Alle Seiten**. Der Wert **Ausstiege** zeigt, welcher Prozentsatz an Besuchern nach dem Aufruf der jeweiligen Seite Deinen Blog verlassen hat. Bei einzelnen Beiträgen können die Ausstiege schon mal bei 80 Prozent liegen. Das passiert etwa, wenn der Beitrag eine bestimmte Frage sehr präzise beantwortet oder weil Du selbst beispielsweise durch eine Link-Liste dort die Ausstiege regelrecht forciert.

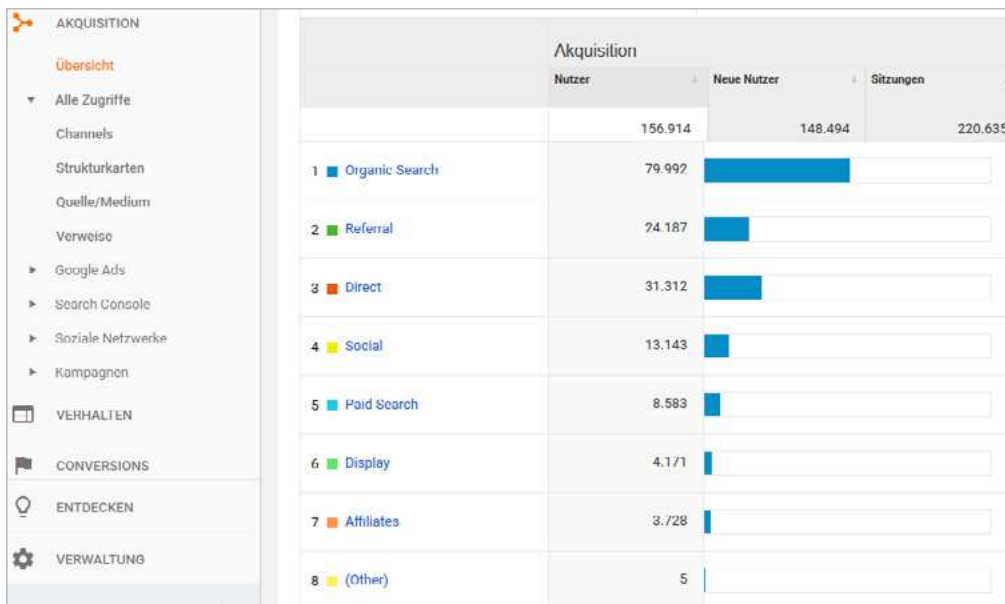
Um Begriffsverwirrungen zu vermeiden: Die Abbruchrate – Bounce Rate – bezeichnet den Anteil der User, die Deinen Blog nach dem Lesen nur einer einzigen Seite wieder verlassen. Die »Ausstiege« zeigen Dir dagegen an, welches die jeweils letzte Seite war, die ein User angesehen hat, bevor er Deinen Blog verlassen hat – unabhängig davon, wie viele Seiten er zuvor bei Dir angesehen hat.

	Seite	Absprungrate	% Ausstiege
		<b>9,65 %</b> Durchsn. für Datenansicht: 9,65 % (0,00 %)	<b>33,93 %</b> Durchsn. für Datenansicht: 33,93 % (0,00 %)
	11.	7,46 %	42,70 %
	12.	8,89 %	27,83 %
	13.	5,98 %	46,79 %
	14.	5,38 %	32,20 %
	15.	4,78 %	43,05 %
	16.	4,33 %	56,86 %
	17.	5,04 %	71,56 %
	18.	5,79 %	68,30 %
	19.	5,38 %	27,50 %
	20.	4,08 %	58,63 %

Absprungrate und Ausstiege, aufgegliedert nach einzelnen Blog-Beiträgen.

### 3. Traffic-Quellen

Jetzt weißt Du, wann Leser gehen. Aber wie haben sie Deinen Blog eigentlich gefunden? Die Herkunft der Besucher ist spannend, weil Du daraus ableiten kannst, wo Du mit mehr Aktivität brachliegende Potenziale ausschöpfen kannst. Und Du kannst identifizieren, welche Kanäle für Deinen Blog besonders gut funktionieren: Ist es eher Facebook oder Twitter? Lohnt sich ein Newsletter oder ein RSS-Feed? Musst Du mehr für Suchmaschinenoptimierung tun, weil zu wenige Besucher über Google zu Dir finden?



Diese Website holt sich viel Traffic über Suchmaschinen.

Die wichtigsten Herkunftsquellen sind:

- **Organic Search:** die normale Suche bei Google & Co.
- **Referral:** Links von anderen Websites auf Dein Blog
- **Direct:** Bookmarks, direkt eingetippte URLs und ähnliches
- **Social:** Verweise bei Facebook, Twitter und Co.
- **Paid Search:** bezahlter Traffic über Anzeigen in Suchmaschinen
- **Email:** Links in E-Mails, typischerweise Dein eigener Newsletter
- **(Other):** zum Beispiel Links aus RSS-Feeds auf

Mit einem Klick auf den jeweiligen Kanal zeigt Google Analytics Dir weitere Details, etwa aus welchen Social-Media-Plattformen genau Deine Besucher kommen.

## 4. Populäre Texte

Eine der wichtigsten Fragen: Welche Blog-Beiträge waren die beliebtesten? In der bereits für die Ausstiege relevanten Tabelle findest Du bei Google Analytics auch die absoluten Abrufzahlen und prozentualen Werte zu jedem einzelnen Deiner Beiträge. Achte aber beim Vergleich mehrerer Beiträge darauf, dass manche vielleicht erst ein paar Tage online sind, andere dagegen schon Monate. Letztere haben dadurch automatisch mehr Zugriffe als die ganz frischen.

Benutzerdefiniert	Seite ?	Seitenaufrufe ?
↳ Benchmarking		<b>777.066</b>
Nutzerfluss		% des Gesamtwerts: 100,00 % (777.066)
Akquisition	<input type="checkbox"/> 11. [blurred]	4.773 (0,61 %)
	<input type="checkbox"/> 12. [blurred]	4.708 (0,61 %)
Verhalten	<input type="checkbox"/> 13. [blurred]	4.666 (0,60 %)
Übersicht	<input type="checkbox"/> 14. [blurred]	4.392 (0,57 %)
Verhaltensfluss	<input type="checkbox"/> 15. [blurred]	4.332 (0,56 %)
↳ Website-Content	<input type="checkbox"/> 16. [blurred]	4.204 (0,54 %)
Alle Seiten		

Analyse der am häufigsten aufgerufenen Blogbeiträge

Konzentriere Dich bei der Auswertung nicht nur auf das Thema des jeweiligen Beitrags als Ursache für dessen Popularität, sondern betrachte auch die Wirkung der Überschrift und des Vorspanns sowie die Textlänge und optische Präsentation, um daraus für künftige Beiträge zu lernen.

Tipp: Wenn Du tiefer in die Analyse einzelner Beiträge einsteigen willst, liefert das WordPress-Plugin [↳ Google Analytics Dashboard By Analytify](#) detaillierte Zahlen zu jedem Beitrag direkt im Backend von WordPress.

## 5. Messen, was andere über Dich denken

Die vielleicht härteste Währung ist »Engagement«, also die Aktivität Deiner Leser, auch »Involvement« oder »Stickyness« genannt. Haben Dein Blog und Deine blogbezogenen Aktivitäten auf Facebook & Co. eine hohe Zahl an Shares, Likes, Mentions, Retweets und Kommentaren, beschäftigen sich Deine Leser intensiv mit dem, was Du schreibst. Gute Werte im direkten Vergleich zu ähnlichen Websites zeigen, dass Du etwas bewegst und Einfluss hast.

Erneut: Beobachte die Steigerungsraten, statt auf absolute Werte zu schauen. Was helfen Dir langfristig 2.500 Facebook-Fans, wenn keine neuen mehr hinzukommen und kaum jemand auf Deine neuen Postings reagiert? Wenige aktive Fans/Freunde sind bei Facebook mehr wert als eine große Zahl passiver Fans, die Du vielleicht irgendwann einmal über ein Gewinnspiel akquiriert hast. Letztere bekommen Deine Postings vermutlich ohnehin längst nicht mehr zu Gesicht, weil mangels Interaktivität dieser User mit Deiner Fanseite die Filterfunktion von Facebook zuschlägt.

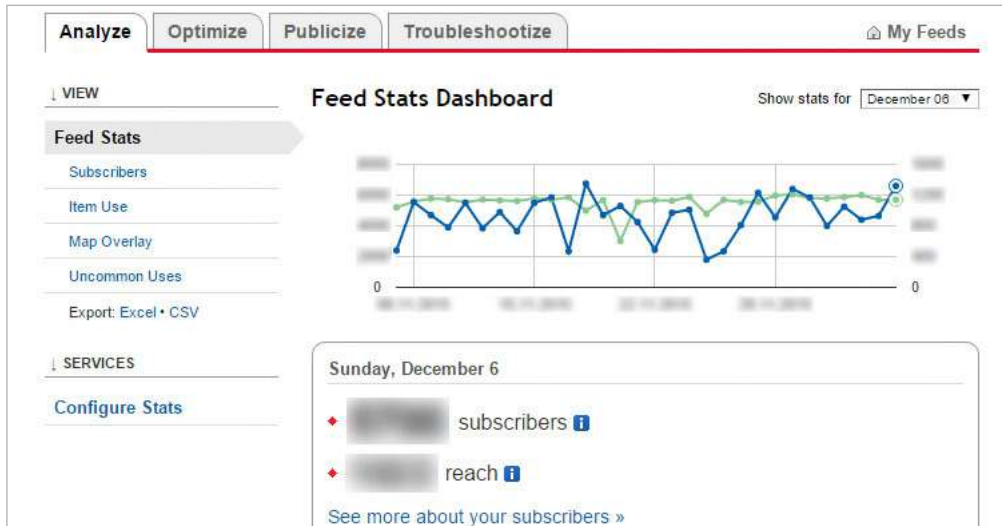
## 6. Links

Wird Dein Blog von anderen Bloggern wahrgenommen, die sich mit ähnlichen Themen beschäftigen? Ein Anhaltspunkt dafür sind Verbindungen auf Social-Media-Kanälen, ein anderer die eingehenden Links. Die Google Webmastertools beispielsweise liefern Informationen darüber, welche Websites auf Deinen Blog verlinken.

## 7. Abonnenten

Ein Aspekt, der in Zeiten von Facebook & Co. gerne übersehen wird: Abonnenten Deines Newsletters und RSS-Feeds sind nicht nur wertvolle Indikatoren für die Treue Deiner Leser, sie sind auch eine Traffic-Quelle, die Du direkt in der Hand hast. Denn: Stellst Du einen neuen Beitrag online, erreichst Du diese Abonnenten über Deinen RSS-Feed direkt. Und verschickst Du einen Newsletter, führen die Links darin ohne Umwege auf Deinen Blog.

Damit Du die Zahl Deiner Feed- und Newsletter-Abonnenten messen und deren Reaktion darauf verfolgen kannst, empfiehlt sich die Nutzung von kostenlosen Diensten, die entsprechende Statistiken bereitstellen. Für RSS-Feeds eignet sich [›Feedburner](#), für Newsletter lohnt sich ein Blick beispielsweise auf [›Mailchimp](#).



Feedburner liefert genaue Statistiken über die Abonnenten Deines RSS-Feeds.

## 7.2 Content-Gruppen: Traffic detailliert aufschlüsseln mit WordPress und Google Analytics

Im vorigen Kapitel hast Du wichtige Kennzahlen für das Tracking kennengelernt. Mit sogenannten Content-Gruppen gehst Du einen Schritt weiter. Du willst zum Beispiel feststellen, wie oft ein Beitrag vom Typ »News« pro Monat aufgerufen wird, weil ein Anzeigenkunde genau in dieser Kategorie Banner kaufen will. Oder Du willst ermitteln, ob Beiträge mit oder ohne Bildergalerie erfolgreicher sind. Oder Du fragst Dich, wie oft Deine vielen, aufwändig geschriebenen Theaterkritiken überhaupt nennenswert gelesen werden.

Die Möglichkeiten sind vielfältig und das Grundprinzip sehr einfach: Du definierst Regeln, nach denen Google Analytics die Seitenzugriffe gruppieren soll. Anschließend kannst Du die Traffic-Zahlen nach diesen individuellen Kriterien aufschlüsseln.

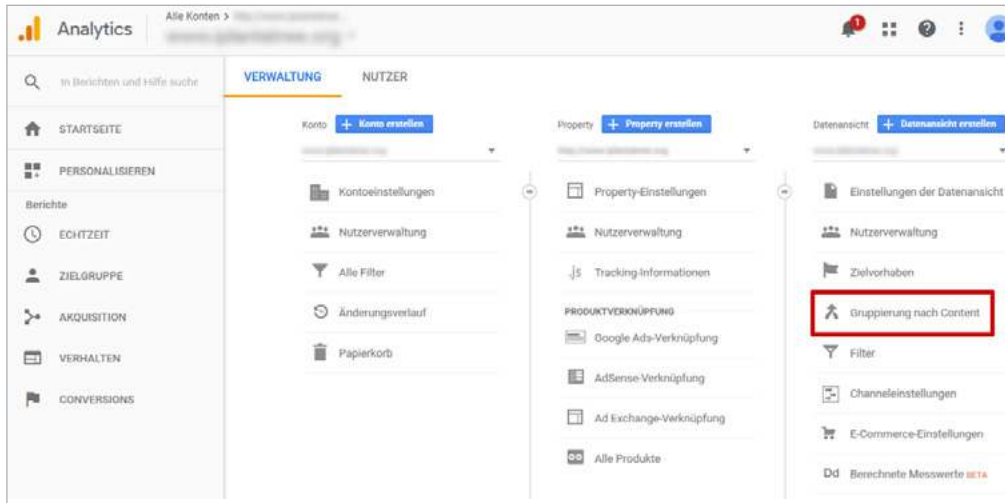
Google Analytics bietet unterschiedliche, teils sehr mächtige, aber auch komplizierte Methoden, um dieses Ziel zu erreichen. Für Blogger sind die sogenannten Content-Gruppen sicherlich die einfachste und bequemste. Je nach Deinen Zielen genügen ein paar Klicks in den Google-Analytics-Einstellungen. Bei komplexeren Anforderungen und für User, die sich ein wenig mit JavaScript und PHP auskennen, lassen sich Content-Gruppen noch individueller einsetzen. Wir beschreiben beide Wege Schritt für Schritt.

### Vorarbeit: Bereiche definieren

Zunächst musst Du entscheiden, was genau Du mit den Content-Gruppen erfassen willst. Für die Schritt-für-Schritt-Anleitung entscheiden wir uns für ein einfaches Beispiel: Wir wollen ermitteln, wie viele Zugriffe die Homepage, die Kategorie **News** und die Kategorie **Bildergalerie** jeweils haben. Die neu zu erstellende Content-Gruppe in Google Analytics nennen wir »Meine Bereiche«.

### Content-Gruppen in Google Analytics anlegen

Klicke in Google Analytics links unten auf **Verwaltung** und wähle anschließend die **Datenansicht** aus, für die Du eine Content-Gruppe erstellen willst. Anschließend klickst Du unten auf **Gruppierung nach Content**.



Anlegen einer neuen Content-Gruppe in Google Analytics

Nun klickst Du auf den roten Button + **Neue Gruppierung nach Content**. Gib Deiner Gruppierung einen aussagekräftigen Namen (in unserem Beispiel »Meine Bereiche«). Jetzt hast Du eine Obergruppe erstellt, von denen Google maximal fünf bereitstellt. Unterhalb einer Obergruppe kannst Du aber beliebig viele Untergruppen anlegen.

Von hier an unterscheiden sich die Methoden: Die Option **Nach Trackingcode gruppieren** ist – ebenso wie **Durch Extraktion gruppieren** – für erfahrene Nutzer mit PHP- und JavaScript-Kenntnissen gedacht. Darauf gehen wir später noch ein.

### Einstellungen der Gruppierung nach Content

**Name**

**Gruppen konfigurieren**

Mithilfe von Gruppierungen nach Content können Sie Ihren Website- oder App-Content in logische Gruppen einteilen und diese Gruppen als primäre Dimensionen in Ihren Berichten verwenden. Sie können Ihren Content mit einer oder mehrerer der unten aufgeführten Methoden gruppieren. [Weitere Informationen](#)

**NACH TRACKING-CODE GRUPPIEREN**

+  Tracking-Code aktivieren

**DURCH EXTRAKTION GRUPPIEREN**

+  Extraktion hinzufügen

**DURCH REGELDEFINITIONEN GRUPPIEREN**

+  Regelsatz erstellen

Die drei Möglichkeiten, Content-Gruppen in Google Analytics zu definieren



Durch Regeldefinitionen gruppieren ist dagegen ohne Eingriff in den Programmiercode von WordPress möglich und damit am einfachsten anzuwenden.

## Gruppieren mit Regeldefinitionen

Klickst Du auf das graue Feld **Regelsatz erstellen**, kannst Du Deine Regeln frei definieren. Mit der Auswahl **Seite** filterst Du nach Elementen in der URI der jeweils aufgerufenen Seite. Die URI (Uniform Resource Identifier) ist der Teil der URL, der nach `http://` und Deiner Domain steht. Die URI von

`http://www.katzenfreunde-suedsachsen.de/ernaehrung/tipps.html`

ist also

`/ernaehrung/tipps.html`

Die Auswahl **Seitentitel** bezieht sich dagegen auf den `title`-Tag im HTML-Code der jeweils aufgerufenen Seite. Oft entspricht das dem Titel des jeweiligen Blogbeitrags – aber das ist abhängig vom jeweiligen Theme, sodass Du prüfen musst, welcher Text bei Deinem Theme im `title`-Tag steht. Rufe dazu einfach den Quelltext einer Seite im Browser auf (in den meisten Browsern: **Strg + U**) und suche nach `<title>`. Innerhalb des öffnenden und schließenden Title-Tags findest Du den gesuchten Text:

`<title>das ist der Text, den Du suchst</title>`

```

1 <!DOCTYPE html>
2 <html lang="de-DE" class="no-js no-svg">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1">
6 <link rel="profile" href="http://gmpg.org/xfn/11">
7
8
9 <title>Immer mehr Katzen ernähren sich vegan 🐾🌱; Katzenfreunde Südsachsen</title>
10 <meta name="robots" content="noindex,follow" />
11 <link rel="dns-prefetch" href="//fonts.googleapis.com" />
12 <link rel="dns-prefetch" href="//s.w.org" />
13 <link href="https://fonts.gstatic.com" crossorigin rel="preconnect" />
14 <link rel="alternate" type="application/rss+xml" title="Katzenfreunde Südsachsen &raquo;" />
15 <link rel="alternate" type="application/rss+xml" title="Katzenfreunde Südsachsen &raquo;" />
16 <link rel="alternate" type="application/rss+xml" title="Katzenfreunde Südsachsen &raquo;" />
17 <script type="text/javascript">
18     window.wpemojiSettings = {"baseUrl":"https://s.w.org/images/core/emoj
19     !function(a,b,c){function d(a,b){var c=String.fromCharCode;1.clearRect(0,0,
20     }</script>
21 <style type="text/css">
22 img.wp-smiley,
23 img.emoji {
24     display: inline !important;

```

Der `<title>`-Tag im HTML-Code eines Blogbeitrags

Du könntest also beispielsweise mit einer **Seitentitel**-Regel, der Option **enthält** und dem Suchtext »vegan« alle Blogbeiträge filtern, deren Titel »vegan« enthält.

Beispiel: Gruppieren Sie alle Beiträge, die »vegan« enthalten.

The screenshot shows a web interface for defining rules. At the top, it says 'DURCH REGELDEFINITIONEN GRUPPIEREN'. Below that, there's a search bar with '1.' and 'vegan'. The main section is 'Regeln definieren'. It has a dropdown for 'Seitentitel', a dropdown for 'enthält', and a text input for 'vegan'. To the right are buttons for '- ODER UND'. Below the dropdown is a list of options: 'stimmt genau überein', '✓ enthält', 'beginnt mit', 'endet mit', 'stimmt mit regulärem Ausdruck überein', 'ist einer von', 'stimmt nicht genau überein', and 'enthält nicht'. There are also buttons for 'Fertig', 'Speichern', and 'Abbrechen'.

Für die Regeldefinition stehen viele Optionen zur Auswahl.

Du kannst natürlich mehrere Definitionen erstellen, um unterschiedliche Gruppen von Inhalten zu differenzieren. Dabei musst Du lediglich darauf achten, dass Analytics die Regeln von oben nach unten durcharbeitet und stoppt, sobald ein Kriterium erfüllt ist. Suchst Du also erst nach einem Titel, der »Katze« enthält, und anschließend nach einem Filter mit »Katzenfutter« im Titel, werden auch alle »Katzenfutter«-Beiträge als »Katze« gezählt, weil bereits die erste Bedingung erfüllt ist.

Auch Kombinationen mehrerer mit **und/oder** verknüpfte Kriterien kannst Du hier definieren.

The screenshot shows the same interface as before, but with two rules defined. The first rule is 'Seitentitel' containing 'test' with the operator 'enthält nicht'. The second rule is 'Seitentitel' containing 'versuch' with the operator 'enthält nicht'. The rules are connected by an 'ODER' operator. There are buttons for 'Fertig' and 'Abbrechen'.

Mehrere Kriterien verknüpfst Du mit und/oder.

Die Vielzahl der Filtermöglichkeiten kann da schnell unübersichtlich werden und zu logischen Fehlern führen. Prüfe also Deine Regeln sehr genau, damit die Auswertung letztlich sinnvolle Werte ergibt.

Zurück zu unserem Beispiel der Sortierung nach **News**, **Bildergalerie** und **Homepage**. Voraussetzung für diese Methode ist, dass sich die Kategorien über die URI identifizieren lassen. News-Beiträge hätten also beispielsweise die URL **http://www.katzenfreunde-suedsachsen.de/news/testnews.html** und Beiträge der Bildergalerie-Kategorie etwa **http://www.katzenfreunde-suedsachsen.de/bildergalerie/galeriebeitragtest.html**.

Eine entsprechende Content-Gruppierung erreichst Du dann mit folgenden Regeln:

Seite enthält nicht /

Seite beginnt mit /news/

Seite beginnt mit /bildergalerie/

Die einzelnen Regel-Definitionen kannst Du leider in der Übersicht aller Regeln in Google Analytics nicht direkt sehen. Dazu musst Du jede Regel zum Bearbeiten einzeln öffnen.



Der fertige Regelsatz in Google Analytics

Bitte beachte aber, dass die Regeln je nach Konfiguration Deines Webspaces und Deiner WordPress-Installation unterschiedlich ausfallen können. Du solltest daher ausführlich testen und alle Varianten berücksichtigen, die eine Rolle spielen. Beispielsweise könnte Deine Homepage sowohl mit als auch ohne abschließendem »/« aufrufbar sein. In diesem Fall müsstest Du die Regel für die Homepage-Erkennung entsprechend abändern, damit Google Analytics beide Varianten berücksichtigt.

Falls die Filterung über die Regeldefinitionen nicht funktioniert, weil weder URI noch Seitentitel die nötigen Unterscheidungsmerkmale bieten, bleibt nur der aufwändigere Weg über den Tracking-Code.

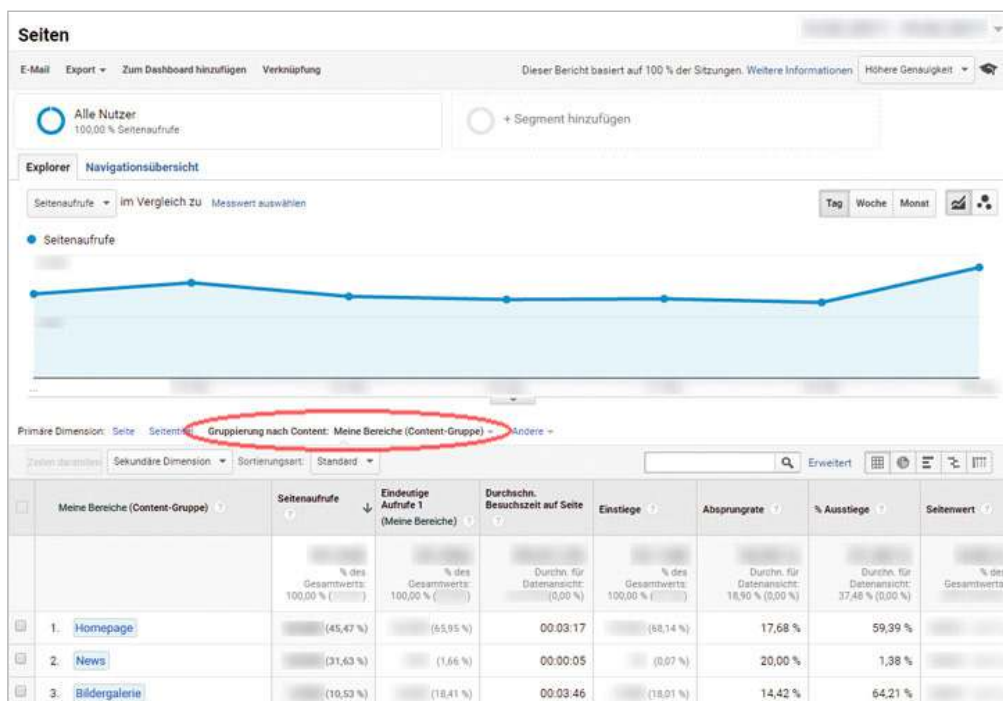
Doch sehen wir uns zunächst an, wie Du die Content-Gruppen bei der Auswertung der Daten von Google Analytics nutzen kannst.

## Content-Gruppen bei der Traffic-Analyse anwenden

Wechsle in Google Analytics in die Ansicht **Verhalten > Websitecontent > Alle Seiten**. Zwischen der Grafik und der Wertetabelle findest Du die Option **Gruppierung nach Content**.

Hier wählst Du nun Deine neue angelegte Content-Gruppe »Meine Bereiche« aus. Die Gruppe wird angezeigt, sobald Analytics ausreichend Daten gesammelt hat. Sie ist also nicht sofort nach Erstellen sichtbar. Spätestens am folgenden Tag sollten aber genügend Daten vorhanden sein.

Jetzt zeigt Google Analytics die Traffic-Zahlen sortiert nach Deinen selbst definierten Content-Bereichen an.



So sieht die Auswertung Deiner Content-Gruppen in Google Analytics aus.

## Content-Gruppen via Trackingcode

Die zweite Methode zum Erstellen von Content-Gruppen setzt zumindest minimale Kenntnisse in JavaScript, PHP und WordPress-Programmierung voraus. Sehr kompliziert ist aber auch diese Methode nicht.

Der Vorteil via Trackingcode: Du bist nicht auf Unterscheidungsmerkmale angewiesen, die in der URI oder im Seitentitel enthalten sind. Stattdessen übergibst Du mithilfe von ein paar Zeilen JavaScript- und PHP-Code in Deinem WordPress-Theme zusätzliche Informationen an Google Analytics. Dort kannst Du die Traffic-Auswertung später genau nach diesen Informationen filtern. Du gibst also quasi jedem Seitenaufruf bei der Zählung eine zusätzliche Markierung mit auf den Weg, die Du dann später zur Auswertung nutzen kannst.

Wie Du im Folgenden sehen wirst, klingt das viel komplizierter, als es ist. Um beispielsweise nach unterschiedlichen WordPress-Kategorien zu filtern, genügt eine einzige Codezeile PHP plus eine Zeile JavaScript im Analytics-Trackingcode.

## Child-Theme anlegen und Analytics-Code einbinden

Zunächst empfiehlt es sich, für die notwendigen kleinen Veränderungen im Theme ein Child-Theme anzulegen ([> siehe Kapitel 2.1](#)).

Wenn Du den Google-Analytics-Code bisher über ein Plugin eingebunden hattest, musst Du dieses Plugin deaktivieren und den Code manuell im Theme einbinden. Dazu kopierst Du den Analytics-Code einfach in den Head-Bereich des HTML-Codes. Typischerweise findest Du den Head-Bereich in der WordPress-Datei **header.php**. Füge den Analytics-Code vor dem HTML-Tag `</head>` ein.

## Content-Gruppen mit Tracking-Code

Zunächst ist der Ablauf identisch zum Anlegen einer Content-Gruppe mit Regeldefinitionen (siehe oben). Statt auf das graue Feld **Regelsatz definieren** klickst Du jetzt aber auf das Feld **Tracking-Code aktivieren**. Da Du bei dieser Methode den Code direkt veränderst, solltest Du ein Grundverständnis für den Analytics-Tracking-Code mitbringen.

Nun siehst du den Tracking-Code, den Du in Dein Theme einfügen musst. Allerdings werden wir diesen Code im nächsten Schritt noch modifizieren, bevor wir ihn

einfügen. Die Anleitung geht davon aus, dass Du das neue Universal Analytics nutzt, sodass die zweite der aufgeführten Code-Zeilen gilt.

### Einstellungen der Gruppierung nach Content

**Name**

**Gruppen konfigurieren**  
 Mithilfe von Gruppierungen nach Content können Sie Ihren Website- oder App Content in logische Gruppen einteilen und diese Gruppen als primäre Dimensionen in Ihren Berichten verwenden. Sie können Ihren Content mit einer oder mehrerer der unten aufgeführten Methoden gruppieren.  
[Weitere Informationen](#)

NACH TRACKING-CODE GRUPPIEREN

**1. Tracking-Code aktivieren**

**Aktivieren**  
  AN

**Index auswählen**  
 Wählen Sie eine Indexnummer aus (1-5).

Ändern Sie den JavaScript-Tracking-Code und fügen Sie eines der folgenden Snippets ein. [Weitere Informationen](#)

**Tracking-Code des allgemeinen Website-Tags (gtag.js):**  
`gtag('set', {'content_group1': 'My Group Name'});`

**Tracking-Code für Universal Analytics (analytics.js):**  
`ga('set', 'contentGroup1', 'My Group Name');`

**Tracking-Code für klassisches Analytics (ga.js):**  
`_gaq.push(['_setPageGroup', 1, 'My Group Name']);`

Die ergänzende Zeile für den Tracking-Code von Google Analytics

Klicke auf **Fertig** und dann **Speichern**, um die Content-Gruppe fertig anzulegen.

## Parameter in WordPress ermitteln

Nun ist Google Analytics bereit, die Daten für die neu angelegte Content-Gruppe zu empfangen. Damit der entsprechende Parameter an Analytics übergeben wird, musst Du den oben bereits angesprochenen Tracking-Code im WordPress-Theme um eine Zeile erweitern.

Allerdings fehlt Dir dazu noch ein wichtiges Element: die zu übergebenden Daten – in unserem Beispiel also der Name der Kategorie, in dem sich der jeweilige Blogbeitrag befindet. Das ist die Stelle, an der ein wenig PHP-Code ins Spiel kommt.

Geh an die Stelle, an der Du in der **header.php** den Analytics-Code eingefügt hast. Direkt über der Zeile `<script>` fügst Du folgende Zeilen ein:

```
<?php
$meinebereiche="";

if(in_category('bildergalerie')) {$meinebereiche='Bildergalerie';}
else if(in_category('news')) {$meinebereiche='News';}
else if(is_home()) {$meinebereiche='Homepage';}
?>
```

Wir bedienen uns hier einer Eigenheit von WordPress, den sogenannten Conditional Tags ([->https://codex.wordpress.org/Conditional\\_Tags](https://codex.wordpress.org/Conditional_Tags)). Das sind PHP-Funktionen, mit denen Du die Eigenschaften des jeweils aktuellen Blogposts ermitteln kannst, also beispielsweise welche Kategorien oder Schlagwörter einem Beitrag zugeordnet sind, ob es sich um die Homepage, eine Kategorie-Übersicht, eine einzelne Seite oder ein Posting handelt und vieles mehr.

In unserem Beispiel verwenden wir zwei Conditional Tags: `in_category()` und `is_home()`.

Wenn der aufgerufene Beitrag der Kategorie mit der Titelform »bildergalerie« zugewiesen ist, gibt `in_category('bildergalerie')` den Wert `true` zurück und erfüllt damit die `if`-Bedingung. Google Analytics bekommt also das Signal, dass es sich um den Aufruf einer Seite im Seitenbereich »Bildergalerie« handelt. Das Gleiche würde auch mit dem Kategorienamen oder der Kategorie-ID funktionieren.

`is_home()` gibt `true` zurück und erfüllt damit die `if`-Bedingung, wenn es sich bei der aufgerufenen Seite um die Homepage handelt.

Es würde zu weit führen, Conditional Tags von WordPress hier ausführlich darzustellen, aber mit ein wenig Einarbeitung kannst Du mit der Vielzahl von möglichen Tags auch recht komplexe und stark differenzierte Content-Gruppen bilden. Interessant dafür könnten beispielsweise auch Conditional Tags wie `is_author()` zur Ermittlung des Autors eines Beitrags sein oder `is_single('beispieltag')` für Posts mit dem Schlagwort »beispieltag« oder `has_post_thumbnail()`, um zu klären, ob dem Beitrag ein Artikelbild zugeordnet ist.

## Ergänzung des Analytics-Codes

Die PHP-Variablen **\$meinebereiche** enthält in unserem Beispiel jetzt einen der Werte **Bildergalerie**, **News** oder **Homepage** oder ist leer, wenn die aufgerufene Seite keinem der drei Bereiche angehört.

Im nächsten Schritt musst Du den Inhalt der Variable nun noch als Parameter an Google Analytics übergeben. Dazu erweiterst Du den Analytics-Code um die folgende Zeile:

```
ga('set', 'contentGroup5', '<?=$meinebereiche?>');
```

**contentGroup5** ist der Parameter-Name, den Google Analytics in dem JavaScript-Code zu Deiner neu angelegten Content-Gruppe definiert hat. Diesen Parameter belegen wir in der Code-Zeile nun mit dem Wert der PHP-Variablen **\$meinebereiche** und schon erfasst Google Analytics, aus welchem Deiner selbst gewählten Bereiche die gerade erfasste Seite stammt.

Der komplette Code für unser Beispiel lautet (statt UA-XXXXXX-XX gibst Du Deinen individuellen Google Analytics Code ein):

```
<?php
$meinebereiche='';
if(in_category('bildergalerie')) {$meinebereiche='Bildergalerie';}
else if(in_category('news')) {$meinebereiche='News';}
else if(is_home()) {$meinebereiche='Homepage';}
?>
<script>
(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
(i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.
insertBefore(a,m)
})(window,document,'script','//www.google-analytics.com/analytics.js','ga');
ga('create','UA-XXXXXX-XX','auto');
ga('set','contentGroup5','<?=$meinebereiche?>');
<em>ga('set','anonymizeIp',true);
ga('send','pageview');
</script></em>
```

Jetzt heißt es ein wenig warten, damit Google Analytics Daten sammeln kann, bevor Du sie in der Traffic-Auswertung sehen kannst.



Hast Du alles richtig gemacht, wirst Du die Segmentierung Deines Contents spätestens nach einigen Stunden in Google Analytics sehen. Beachte aber, dass die Gruppierung natürlich nicht rückwirkend funktioniert. Analytics zeigt die Content-Gruppen-Auswertung also erst an, wenn es entsprechende Daten sammeln konnte.

Die Auswertung der gesammelten Daten funktioniert genauso wie oben bereits beschrieben. Du kannst sogar die unterschiedlichen Methoden der Content-Gruppierung kombinieren – Trackingcode, Extraktion (Profi-Option für Filter über sogenannte Regular Expressions) und Regeldefinitionen. Google Analytics arbeitet die drei Varianten in genau dieser Reihenfolge ab und stoppt daher, sobald in diesem Ablauf das erste Kriterium erfüllt ist.

# 8 WordPress Multisite

Mit [WordPress Multisite](#) kannst Du beliebig viele WordPress-Websites in einer Installation zentral verwalten. Themes und Plugins musst Du nur einmal im Netzwerk installieren und kannst sie dann für alle Websites einsetzen. Die Administration ist dadurch viel effizienter. Praktisch ist Multisite auch zum Testen neuer Funktionen oder Themes.

Bei der Einrichtung hast Du die Wahl, ob die Netzwerk-Seiten Subdomains, Verzeichnisse oder verschiedene Domains verwenden sollen. Dies kann später nicht geändert werden.

## 8.1 Multisite mit Verzeichnissen und Subdomains

### Schritt 1: WordPress manuell installieren

Der STRATO AppWizard eignet sich perfekt für die Installation einzelner Blogs. Für ein Netzwerk von Blogs musst Du WordPress jedoch manuell installieren. Wie das funktioniert, haben wir in [Kapitel 1: Voraussetzungen](#) beschrieben.

### Schritt 2: Multisite aktivieren

Um Multisite zu aktivieren, musst Du nach der Installation die Datei **wp-config.php** anpassen. Falls diese im Webservice nicht zu sehen ist, hat WordPress sie noch nicht erstellt. Lösung: In WordPress einloggen und dann die FTP-Verbindung neu aufbauen. Lade die **wp-config.php** herunter und gib oberhalb von `*/ * That's all, stop editing! Happy blogging. */` folgende Zeile ein:

```
define('WP_ALLOW_MULTISITE', true);
```

Anschließend lädst Du die Datei wieder hoch und ersetzt die Original-Datei im Webservice.

### Schritt 3: Netzwerk aktivieren

Wenn Du das Backend aktualisierst, erscheint unter Werkzeuge nun der Eintrag Netzwerk-Einrichtung. Unter »Adressen der Websites in deinem Netzwerk« kannst Du zwischen Subdomains und Unterverzeichnissen wählen:

**Adressen der Websites in deinem Netzwerk**

Bitte wähle, ob die Websites in deinem WordPress-Netzwerk Subdomains oder Unterverzeichnisse verwenden sollen. Dies kann später nicht geändert werden.

Du benötigst einen Wildcard DNS-Eintrag, wenn du die Funktionalität des Virtual Host (Subdomain) verwenden möchtest.

Subdomains wie  und

Unterverzeichnisse wie  und

Bei beiden Varianten werden nach einem Klick auf den Installieren-Button zwei Code-Schnipsel angezeigt: Der erste wird an der gleichen Stelle wie oben in die `wp-config.php`, der zweite Block in die `.htaccess` eingefügt.

Bitte sichere beide Dateien vorab und achte darauf, die `.htaccess` im Hauptverzeichnis abzulegen!

**Aktivieren des Netzwerks**

Vervollständige die folgenden Schritte, um die Funktionen zum Erstellen eines Netzwerks von Websites zu aktivieren.

Vorsicht: Wir empfehlen dir, die existierenden Dateien `wp-config.php` und `.htaccess` zu sichern.

1. Füge folgendes zur deiner `wp-config.php` Datei hinzu, in `/mnt/.../htdocs/` oberhalb der Zeile die `/* That's all, stop editing! Happy blogging. */` enthält:

```
define('MULTISITE', true);
define('SUBDOMAIN_INSTALL', true);
define('DOMAIN_CURRENT_SITE', '...');
define('PATH_CURRENT_SITE', '/');
define('SITE_ID_CURRENT_SITE', 1);
define('BLOG_ID_CURRENT_SITE', 1);
```

2. Füge folgendes zu deiner Datei `.htaccess` in `/mnt/.../htdocs/` hinzu und ersetze andere WordPress-Regeln dadurch:

```
RewriteEngine On
RewriteBase /
RewriteRule ^index\.php$ - [L]

# add a trailing slash to /wp-admin
RewriteRule ^wp-admin/$ wp-admin/ [R=301,L]

RewriteCond %{REQUEST_FILENAME} -f [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^ - [L]
RewriteRule ^(wp-(content|admin|includes).*) $1 [L]
RewriteRule ^(.*)\.php$ $1 [L]
RewriteRule . index.php [L]
```

Nach Abschluss dieser Schritte ist dein Netzwerk aktiviert und eingerichtet. Dann musst du dich erneut anmelden. [Anmelden](#)

Bei der Netzwerk-Einrichtung sagt WordPress, welcher Code wo eingefügt werden muss.

Nach einem erneuten Login ist das Netzwerk fertig eingerichtet. Wenn Du bei der Einrichtung Unterverzeichnisse ausgewählt hast, kannst Du gleich anfangen die Websites erstellen. Klicke dazu oben links neben dem WordPress-Logo auf **Meine Websites > Netzwerkverwaltung > Websites > Neu hinzufügen**.

#### Schritt 4: Subdomains erstellen

Hast Du Dich für die Option Subdomains entschieden, musst Du im **STRATO Kunden-Login** erst noch Subdomains für die einzelnen Websites anlegen. Über **Domains > Domainverwaltung > Verwalten > Subdomain anlegen** ist das schnell erledigt.

Um abschließend die Websites zu erstellen, gehst Du wie bei den Unterverzeichnissen vor: **Meine Websites > Netzwerkverwaltung > Websites > Neu hinzufügen**. Als **Website-Adressen (URL)** gibst Du die oben erstellten Subdomains an.

### Neue Website erstellen

Website-Adresse (URL)  Nur Kleinbuchstaben (a-z), Ziffern und Bindestriche sind erlaubt.

Titel der Website

Sprache der Website  ▼

Administrator E-Mail-Adresse

Es wird ein neuer Benutzer angelegt, sofern nicht schon ein Benutzer mit dieser E-Mail-Adresse vorhanden ist. Der Benutzername und ein Link zum Festlegen des Passworts werden an diese E-Mail-Adresse gesendet.

In der Netzwerkverwaltung kannst Du bequem neue Websites erstellen.

## WordPress Multisite: effizient bei vielen Blogs

WordPress Multisite ist ein mächtiges Tool, um beliebig viele Blogs und Websites zu administrieren. Als sogenannter Super-Administrator kannst Du Themes und Plugins zentral installieren, aktualisieren und für einzelne Websites aktivieren. Wird ein Benutzer auf Website-Ebene angelegt, hat er – wie bei separaten Installationen auch – nur dort entsprechende Rechte.

Wie die Anbindung unterschiedlicher ›[STRATO Domains](#) funktioniert, erfährst Du im folgenden Abschnitt.

## 8.2 Domain Mapping mit WordPress Multisite

Bei WordPress Multisite können Websites des Netzwerks nicht nur Unterverzeichnissen oder Subdomains zugewiesen werden, sondern auch anderen Domains. Für die folgenden Schritte benötigst Du daher ein [STRATO Hosting-Paket](#) mit mindestens zwei Domains.

### Schritt 1: WordPress installieren und Multisite aktivieren

Als Erstes installierst Du WordPress manuell und aktivierst die Multisite-Funktion und das Netzwerk wie in [Kapitel 8.1](#) beschrieben (Schritte 1 bis 3). Wähle im dritten Schritt als Adressen der Websites unbedingt Unterverzeichnisse:

**Adressen der Websites in deinem Netzwerk**

Bitte wähle, ob die Websites in deinem WordPress-Netzwerk Subdomains oder Unterverzeichnisse verwenden sollen. Dies kann später nicht geändert werden.

Du benötigst einen Wildcard DNS-Eintrag, wenn du die Funktionalität des Virtual Host (Subdomain) verwenden möchtest.

Subdomains wie  und

**Unterverzeichnisse** wie  und

Um Probleme beim Einloggen oder mit Berechtigungen zu vermeiden, fügst Du unterhalb des von WordPress vorgegebenen Codes zusätzlich folgende Zeile in die `wp-config.php` ein:

```
define('COOKIE_DOMAIN', '');
```

### Schritt 2: Websites hinzufügen

Sobald das Netzwerk aktiviert ist, erstellst Du die einzelnen Websites: **Meine Websites > Netzwerkverwaltung > Websites > Neu hinzufügen**. Als »Website-Adresse (URL)« kannst Du erstmal irgendwas eintragen – das wird eh gleich wieder geändert (Schritt 3), weil Du ja kein Unterverzeichnis, sondern eine andere Domain nutzen willst.

**Neue Website erstellen**

Website-Adresse (URL)   
*Nur Kleinbuchstaben (a-z), Ziffern und Bindestriche sind erlaubt.*

Titel der Website

Sprache der Website

Administrator E-Mail-Adresse

Es wird ein neuer Benutzer angelegt, sofern nicht schon ein Benutzer mit dieser E-Mail-Adresse vorhanden ist. Der Benutzername und das Passwort werden an die angegebene E-Mail-Adresse gesendet.

[Website erstellen](#)

Beim Erstellen neuer Websites kannst Du als Adresse zunächst nur ein Verzeichnis angeben.

### Schritt 3: Website-Adressen eingeben

Hast Du die Websites erstellt, musst Du nun die richtigen Adressen eingeben. Dafür gehst Du in die Website-Übersicht (**Websites > Alle Websites**). Bewegst Du den Mauszeiger über einen Eintrag, erscheint der jeweilige **Bearbeiten**-Link. Hier kannst Du eine beliebige **>STRATO Domain** aus Deinem **>Hosting-Paket** eingeben.

**Website bearbeiten: Katzenblog**

[Ansehen](#) | [Dashboard](#)

Info | Benutzer | Themes | Einstellungen

Website-Adresse (URL)

Registriert

Zuletzt aktualisiert

Attribute

- Öffentlich
- Archiviert
- Spam
- Gelöscht
- Erwachseneninhalte

[Änderungen speichern](#)

Tipp: Falls Du Dich nicht mehr einloggen kannst, liegt das wahrscheinlich an veralteten Cookies. Schließe in dem Fall einfach den Browser, lösche den Cache und versuche es erneut.

## Fertig!

Wenn alles erledigt ist, kannst Du Dich mit Deinen »Super-Admin«-Zugangsdaten (für das gesamte Netzwerk) in die Backends der einzelnen Websites einloggen. Wenn Du im Backend einer Website einen Benutzer anlegst, hat dieser auch nur dort Zugriff. Viel Spaß mit Multisite!



# Schlusswort

Bist Du startklar? Steckst Du vielleicht schon mitten in Deinem WordPress-Projekt? Dann hoffen wir, dass unser E-Book »WordPress für Fortgeschrittene« für Dich ein guter Ratgeber ist.

Übrigens: Über neue WordPress-Features und Trends informieren wir Dich in unserem [STRATO Blog](#). Hier stellen wir auch regelmäßig interessante und erfolgreiche Blogs vor. Schau doch mal vorbei und hol Dir Anregungen für Deinen eigenen Blog.

Schließlich fragen wir uns: Was ist aus Deinem Projekt geworden? Wir sind sehr gespannt auf Dein Feedback: Besuche uns auf unserem [Blog](#), auf [Facebook](#) oder [Twitter](#).

Frohes Bloggen und viele Grüße

Dein STRATO Team