WordPress-teman

– terminologi och uppbyggnad

Christa Hannuksela

Examensarbete

Informations- och mediteknik

2015

EXAMENSARBETE

Högskolan Arcada

Utbildningsprogram:	Informations- och medieteknik
Identifikationsnummer:	4936
Författare:	Christa Hannuksela
Arbetets namn:	WordPress-teman – terminologi och uppbyggnad
Handledare (Arcada):	Johnny Biström

WordPress har, sedan det kom ut år 2003, vuxit i popularitet till att nu, år 2015, vara det överlägset mest populära webbpubliceringssystemet. Det har därför blivit relevant att känna till hur man skapar ett utseende för en WordPress webbsida.

Det enklaste sättet att gå till väga är att skapa ett barntema, men det finns också startteman som man kan använda som grund för att sedan bygga upp ett önskat utseende. Ett ytterligare alternativ är att bygga upp filstrukturen och innehållet själv. I samtliga fall är det en fördel att man känner till den terminologi som WordPress.org använder i anknytning till teman.

Av de obligatoriska filerna som måste finnas med i ett tema som inkluderas i WordPress.org:s temakatalog finns det två filer som särskiljer sig från resten, nämligen style.css och functions.php. Style.css innehåller information om temat i en kommentar i början av filen, medan functions.php innehåller funktioner som bidrar med utökad funktionalitet till webbsidan.

Resten av de obligatoriska filerna är .php-filer som kallas mallfiler (Template Files). Dessa genererar källkoden. Vissa mallfiler genererar en del av ett HTML-dokument, andra innehåller alla komponenter som behövs för att generera ett fullständigt HTML-dokument. De senare kallas för sidmallar (Page Templates). Här förekommer en del förvirring, eftersom WordPress har egna termer för de mallfiler som genererar olika typer av sidor, så som arkivmall och sökmall, medan jag upplever att ordet sidmall skulle kunna användas för att beskriva alla filer som innehåller de komponenter som behövs för att skapa ett fullständigt HTML-dokument.

Mallfilerna innehåller PHP- och HTML-kod. En stor del av PHP-koden består av WordPress egna funktioner: Malltaggarna (Template Tags) utför åtgärder, letar fram information ur databasen eller letar efter mallfiler medan villkorstaggarna (Conditional Tags) används som villkor i if-satser. Slingan (The Loop) består av både malltaggar och villkorstaggar och behövs för att WordPress skall kunna visa olika sidor, inlägg,

sökresultat o.s.v.

Den praktiska delen av mitt arbete har bestått av att jag byggt en grund för ett tema som uppfyller WordPress temagranskares krav. Min grundliga genomgång av terminologin bidrog till att detta arbete var enkelt.

Nyckelord:	WordPress-teman, PHP, HTML, sidmallar, malltaggar, mallfiler
Sidantal:	49
Språk:	svenska
Datum för godkännande:	27.5.2015

DEGREE THESIS	
Arcada University of Applied Sciences	
Degree Programme:	Information- and Media Technology
Identification number:	4936
Author:	Christa Hannuksela
Title:	WordPress Themes – terminology and construction
Supervisor (Arcada):	Johnny Biström

Abstract:

WordPress has, since its introduction in 2003, become the most popular CMS used by websites. It is therefore of relevance to know how to create an appearance of one's own for a WordPress website.

The easiest way is to create a Child Theme to another theme, but there are also Starter Themes that are meant to be used as something to build from. Lastly one can create one's own file structure and content. In all of these cases it is an advantage to know the terminology which is used in connection with WordPress themes.

Among the files that have to be included in a theme that qualifies for submission to the WordPress.org Theme Directory, two stand out from the rest: functions.php and style.css. The function file consists of functions which add functionality to the theme, while the style file contains information about the theme in a comment at the head of the file.

The rest of the files that have to be included are so called Template Files. These are .php files that generate the source code for the web pages. Some of these files generate a part of a web page, while others generate a complete web page. The latter ones are called Page Templates. Although WordPress.org differentiates between Page Templates that generate pages and Template Files that generate archives, search results etc. I prefer to use the term Page Template for all of them because they all generate complete web pages.

The Template Files contain PHP and HTML. A great part of the PHP code consists of WordPress functions: The Template Tags perform actions, look up information from the database or include the contents of Template Files, while Conditional Tags are used as conditions in if-statements. The Loop that loops through the result of a query and produces the results – single pages, posts, search result etc. – consists of both Template Tags and Conditional Tags.

In the practical part of my work I have created what could be considered my own Starter Theme. The good understanding I had gotten from reading the WordPress Codex and the Theme Handbook made my work easy.

Keywords:	WordPress themes, PHP, HTML, Page Templates, Templates, Template Tags
Number of pages:	49
Language:	Swedish
Date of acceptance:	27.5.2015

INNEHÅLL

FIGURER

WORDPRESS-TERMINLOGI

barntema ……..... Child Theme

inkluderande taggar ……... Include Tags

förältertema …….. Parent Theme

mall ……...… Template

mallfil …… Template File

mallhierarki …….. Template Hierarchy

mallnamn ……..…. Template Name

malltagg ……. Template Tag

sidmall …… Page Template

tillägg …… Plugin

villkorstagg …… Conditional Tag

ÖVRIG TEKNISK TERMINOLOGI

sidfot ……. footer

sidhuvud ……..... header

stilmall …….. Cascading Styles Sheet (CSS)

sökväg ……..… file path

uppställning ……..… array

1 INLEDNING

WordPress har sedan det lanserades i maj år 2003 (WordPress:l) vuxit i popularitet, till att det nu våren 2015, anses vara det mest populära CMS:et. (W3Techs, [2015])[1] (Wappalyzer, [2015])[2]. Med CMS avses ett system, som används för att hantera, ändra alternativt korrigera och publicera information (Wikipedia, 2015). I dagens läge används oftast uttrycket CMS för att beskriva webbpubliceringssystem.

WordPress är ett traditionellt verktyg för webben i den bemärkelsen att det drivs med programmeringsspråket PHP och databashanteraren MySQL.

1.1 Syfte och tillvägagångssätt

Avsikten med mitt arbete är beskriva hur man skapar ett tema, d.v.s. ett utseende, för WordPress. Jag kommer att beskriva vad WordPress-teman är för någonting – vad de består av och hur de är strukturerade, samt olika tillvägagångssätt för hur man kan skapa ett skräddarsytt utseende för en webbsida som drivs med WordPress. I min presentation av hur teman är strukturerade strävar jag till att ge en god grundförståelse för den terminologi som används av WordPress.

Jag kommer också att inkludera en komparativ del där jag går igenom några aktuella WordPress-teman för att eventuellt kunna dra några slutsatser om vad som är typiskt för teman just nu våren 2015.

---

[1] WordPress används på 23,8% av alla de webbsidor som W3Techs har besökt. Av de CMS som W3Techs kan känna igen är WordPress andel 60,3%. W3Techs krälar (crawl) de 10 miljoner mest besökta sidorna som finns listade på Alexa.com.

[2] Wappalyzer anger att WordPress används på 64% av de webbsidor människor har besökt under det senaste halvåret. Wappalazer är ett webbläsartillägg som samlar information via sina användare (Wappalyzer, u.å.)

Den praktiska delen av mitt arbete går ut på att jag bygger en grund för ett tema, d.v.s. jag skapar den filstruktur och det innehåll som behövs för att tema skall kunna godkännas av temagranskaren (Theme Review), men jag kommer inte inom ramarna för detta arbete att framställa en fungerande estetisk helhet. I min genomgång av det praktiska arbetet kommer jag att referera till det som jag behandlat i tidigare kapitel samt gå igenom sådant som ytterligare behövs för att kunna bygga ett fungerande tema.

Det jag bygger skall uppfylla de krav som WordPress temagranskare (Theme Review) ställer. Slutprodukten skall gå att använda till ett tema som är ämnat för självpublicering. Med självpublicering avser jag den slags publiceringsverksamhet som är vanligt för en privatperson så som en blogg och/eller en portfolio.

1.2 Källor och litteratur

WordPress utvecklas och uppdateras konstant[3], vilket syns i hur ofta nya versioner kommer ut (WordPress:l), och därför blir dess egna dokumentations- och instruktionssidor de bästa källorna till vilka regler och terminologi som gäller vid utvecklingen av teman. Mina huvudsakliga källor kommer därför att vara WordPress.org:s Codex-sidor (WordPress:m), samt WordPress.org:s Theme Handbook (WordPress:r), hädanefter refererade till som Codexen och Handboken. Det verkar som att WordPress håller på att gå över till den senare och Codexen kommer antagligen att avvecklas i något skede. I skrivande stund, våren 2015, finns det ändå inga tydliga tecken på detta i Codexen och det finns inte heller några direkta länkar till Handboken på WordPress.org:s webbsidor. Eftersom jag upplever att Handboken är bättre strukturerad än Codexen har jag valt att ta med den, även om jag kommer att hänvisa till Codexen i de fall jag tycker att den är tydligare. En brist med WordPress egna sidor är att det varken finns någon datumanmärkning för när en sida har skapats eller uppdaterats senast. Det här gör det svårt att utöva källkritik gentemot texter som handlar om WordPress, eftersom det inte går att avgöra om skribenterna har missförstått något eller om det har skett förändringar i Codexen.

Eftersom WordPress är ett populärt verktyg finns det också en mängd med blogginlägg, forum och videon med diskussioner och tutorialer som går igenom hur man skapar och hanterar en WordPress-webbsida. De internetkällor som jag använder mig av har jag ändå oftast funnit via WordPress.org, alternativt har jag bedömt att den information som presenteras i källan är trovärdig och saknar motsvarighet i Codexen och Handboken.

Som en följd av WordPress popularitet finns det också en mängd med böcker som handlar om verktyget. Många av dessa verk går igenom hur man skapar en webbsida och betraktar WordPress utifrån dess administrationspanel istället för att diskutera dess filer och filinnehåll, vilket gör verken irrelevanta för min studie.[4] Därtill förekommer problematik vad gäller böckers aktualitet – De senaste verken jag lyckades hitta som handlar om att bygga WordPress-teman utkom år 2013.[5]

Jag har haft svårigheter med att hitta akademiska texter som handlar om WordPress – de facto har jag inte hittat ett enda verk utöver arbeten av kandidatnivå som åtar sig publiceringsverktyget. Det finns säkert akademiska texter som behandlar WordPress, men eftersom ämnet för min studie är temautveckling så ser jag de ändå som trovärdigt att detta ämne inte berör akademiska arbeten utöver kandidatnivå.[6]

---

[4] Som exempel kan nämnas *WordPress For Dummies*-serien.

[5] Sabin-Wilson, L. (2013) *WordPress Web Design for Dummies*. 2nd ed.
McCollin, R. & Blakeley Silver, T. (2013) *WordPress Theme Development Beginner's Guide*.

[6] För tillfället är det endast de finska högskolorna som har en gemensam webbportal, Theseus.fi, för examensarbeten. Eftersom det är obligatoriskt för studeranden vid dessa instanser att lägga upp sina examensarbeten på Theseus gör det att den ger en bra överblick av vad andra har skrivit om och också över den allmänna kvalitén på examensarbeten. Visserligen har en del andra nordiska högskolor och universitet egna portaler för examensarbeten, men samtliga finns på instansernas egna webbsidor vilket gör dem svåra att hitta. Det här är skälet till att alla examensarbeten jag kommer att nämna är finska.

1.3 Terminologi

Jag har valt att inkludera en terminologilista med översättningar i början av mitt arbete. Då jag första gången i ett kapitel använder mig av en term som är relaterad till WordPress hänvisar jag alltid till den engelska termen inom parentes. Detta eftersom jag själv har översatt en stor del av den terminologi som jag använder mig av och inte vill att det skall uppstå någon oklarhet kring dess engelska ursprung.[7] Ifall jag inte ännu har behandlat en term, hänvisar jag också till i vilket kapitel jag kommer göra det.

Då det kommer till ordet **sidhuvud** används det här i tre olika bemärkelser:

1. som en kommentar i början av en fil så att WordPress kan identifiera filen
2. som sidhuvudet till ett genererat HTML-dokument. I WordPress brukar ofta HTML-dokumentet vara uppdelat i flera delar varav en heter header.php. Denna fil innehåller ofta mer än det som finns inom HTML-dokumentets `<head>`-taggar
3. som en del av layouten. Sidhuvudet befinner sig oftast någonstans kring webbsidans övre kant och består i den mest avskalade versionen endast av namnet på webbplatsen i fråga

Ifall jag anser att ett förtydligande av vilkendera termen jag använder mig av kommer jag att klargöra det i texten.

[7] Av de slutarbeten jag har tittat på görs inga vidare försök till att gå igenom den terminologi som används vid byggandet av teman och tillägg för WordPress. Därtill har jag funnit få försök till översättningar. T.ex. i Myllykangas 2014 förekommer termen ”sivupohja” utan att det görs någon referens varken till Template File eller Page Template, medan Poranen 2015 visserligen gör en referens med förtydligar inga andra termer. Kallinki & Pasanen 2013 använder det engelska ordet ”template”. Mäkiniemi 2014 har till skillnad från ovannämnda refererat till de engelska orden, men arbetet är en snabb översikt över hur WordPress fungerar, så det berör inte den terminologi som har att göra med byggandet av teman.

1.4 Syntax

Under det senaste året har Automattic, d.v.s. företaget som driver bloggportalen WordPress.com och som styr utvecklingen av WordPress, övergått från att använda sig av syntaxen:

```
if ( $villkor || villkor() ) {
      sats();
}
```

till att använda sig av

```
if ( $villkor || villkor() ) :
      sats();
endif;
```

i sina WordPress-teman.

Skälen till detta nämns ingenstans, men det kan bero på att den senare anses vara mer lättläst för designers (Davis, 2011). Jag har valt att följa WordPress exempel och alla längre partier av kod kommer använda sig av den senare versionen. En stor majoritet av den PHP-kod jag kommer uppvisa är inte unik, utan det är obligatoriskt, eller åtminstone rekommenderat att den skall inkluderas i ett tema. Jag kommer precisera vilket är fallet i mina exempel.

1.5 Avgränsningar

Jag har utgått ifrån att läsaren känner till vanliga begrepp som används vid byggandet av webbsidor såsom HTML, CSS, PHP och JavaScript. Därutöver har jag utgått ifrån att läsaren är bekant med hur man installerar WordPress på en server, samt att denne känner till hur man använder WordPress för att skapa blogginlägg (Posts) och enskilda sidor (Pages).

Jag kommer inte att gå igenom WordPress API i någon större utsträckning utan vill endast ge den information som någon skulle behöva för att förstå sig på den terminologi och den filstruktur som är en del av WordPress, och som man måste känna till för att kunna lösa vanliga problem som kan uppstå då man vill påverka utseendet på en

WordPress- webbsida. Jag kommer inte heller att gå igenom alla de krav som ställs på teman som inkluderas i temakatalogen (Theme Directory, se kapitel 2), eftersom dessa innefattar allt från funktionalitet till krav som ställs på PHP- och CSS-koden.

2 WORDPRESS-TEMAN

WordPress-teman är till för att skapa en utseendemässig ram på en WordPress-baserad webbsida för det material någon vill publicera. Teman introducerades i.o.m. WordPress 1.5 (WordPress:j). En WordPress-webbsida fungerar inte om den saknar tema, d.v.s. om temamappen (`wp-content/themes`) är tom.

WordPress.org upprätthåller Temakatalogen (Theme Directory). De teman som finns med i katalogen har gått igenom en granskning för att se till att alla teman som ingår uppfyller de krav som ställs på koden, funktionaliteten och filerna. (WordPress:n)

2.1 Ett anpassat utseende för en WordPress-sida

Det finns flera alternativ till hur man kan sätta igång med att skapa ett eget utseende för en webbsida som drivs med WordPress. Nedan har jag gjort en uppdelning i tre.

2.1.1 Barnteman (Child Themes)

Möjligheten att skapa barnteman (Child Themes) så som de görs i nuläget introducerades i.o.m. WordPress 2.7 (Chandler, 2008) (Eastaugh, 2009). Att skapa ett barntema är inte liktydigt med att skapa ett tema, enär ett barntema inte fungerar utan tillgång till ett föräldertema, d.v.s. föräldertemat måste finnas i den aktuella WordPress-installationens `wp-contents/themes`-mapp. (WordPress:a) Nästan vilket som helst WordPress-tema kan fungera som föräldertema.

Barntemat ärver föräldertemats egenskaper samtidigt som barntemats skapare kan göra egna utseendemässiga och funktionella förändringar. En av fördelarna med att bygga ett

barntema är att man redan från början har en fullt fungerande webbsida, vars utseende och funktionalitet man kan styra över, utan att för den delen behöva rädslas för att man i misstag skulle ha sönder viktig funktionalitet som är del av det tema man arbetar utifrån. Att skapa ett barntema istället för att ändra på innehållet i ett tema, vars utveckling man inte bestämmer över, försäkrar att alla de förändringar man utfört inte går förlorade om (förälder)temat uppdateras. (WordPress:a)

För att skapa ett barntema behövs ett tema att arbeta utifrån, d.v.s. föräldertemat, samt en mapp för barntemat. Barntemats mapp bör innehålla stilfilen `style.css` och funktionsfilen `functions.php`.

Figur 1 Exempel på filstruktur för ett simpelt barntema

Stilmallsfilen behöver ett sidhuvud där barntemats namn framkommer samt föräldertemats mappnamn anges. WordPress rekommenderar att barntemats mapp skall namnges `{föräldertemats-mapp-namn}-child.`

```
/*
Theme Name:       Mitt första barntema
Template:         twentyfifteen
*/ 8
```

Exempel på småskaligt sidhuvud i ett barntema som heter ”Mitt första barntema”. Föräldertemat är Twenty Fifteen.

Funktionsfilen behövs för att inkludera föräldertemats stilmallsfiler och för att utföra eventuella förändringar i webbsidans funktionalitet. För att inkludera föräldertemats stilfil måste man använda sig av WordPress-funktionen `wp_enqueue_style(`

[8] Det finns betydligt fler taggar än dessa två, se till exempel (WordPress:a)

`$handle, $src, $dep, $ver, $media )`

(https://codex.wordpress.org/Function_Reference/wp_enqueue_style) genom att inkludera följande, eller en liknande, kodsnutt i funktionsfilen:

```
<?php
function theme_name_add_styles() {
    wp_enqueue_style( 'parent-style-name',
get_template_directory_uri() . '/style.css' );
}

add_action( 'wp_enqueue_scripts', 'theme_name_add_styles' );
?>
```
[9]

Ifall föräldertemat har fler stilmallsfiler måste också dessa inkluderas med `wp_enqueue_style()` innanför den av temautvecklaren definierade funktionen. (WordPress:a)

Barntemats mapp skall placeras i WordPress-installationens `wp-contents/themes`-mapp och därefter kan barntemat aktiveras via administrationspanelen.

Skaparen till barntemat kan också lägga till egna mallfiler (Template Files, se kapitel 3.3.1) alternativt ändra strukturen på de mallfiler som används av föräldertemat samt ändra på och/eller lägga till funktionalitet till temat. Det här innebär att det går att göra fler förändringar till en webbsidas utseende än dem som styrs av stilmallsfilerna.

Barnteman kan, ifall de klarar av temagranskningen, inkluderas i WordPress temakatalog.

2.1.2 Startteman (Starter Theme)

Tanken med ett starttema är att den som vill skapa ett eget tema skall ha en fungerande grund att arbeta utifrån. Att använda sig av ett starttema kan eventuellt snabba upp skaparprocessen, men det kan också fungera som en introduktion till hur man bygger

[9] `add_action()` och `wp_enqueue_scripts()` är WordPress-funktioner, `wp_enqueue_scripts()` är en händelsekrok (Action Hook) medan `add_action()` lägger till den av tema- eller tilläggsutvecklaren definierade funktionen till kroken i fråga. Se kapitel 3.2.2

teman. Eftersom startteman brukar uppfylla WordPress krav så behöver den som arbetar utifrån ett starttema inte fundera på om den har glömt att lägga till något – det är sannolikare att hen i misstag har tagit bort någonting.

Det finns ett flertal startteman varav Automattics egna, Underscores (_s, http://underscores.me), antagligen är det mest kända. Dessutom är det sannolikast att Automattic ser till att dess egna teman överensstämmer med de gällande standarderna för WordPress-teman. Andra exempel på startteman är Bones (https://github.com/eddiemachado/bones) och Sage (https://github.com/roots/sage).

Många temautvecklare skapar sig med tiden ett eget grundtema som de sedan utgår ifrån då de bygger nya teman (Rooney, 2013). Med tiden går det sedan att förbättra sin kod och avlägsna partier som eventuellt har föråldrats.

2.1.3 Att börja från noll

I allmänhet, då man skapar WordPress-teman, eller någon annan kod börjar man inte helt från noll, utan det är allmänt bruk att använda sig av andras kod som material. Skiljelinjen mellan att använda någonting som någon annan skapat, t.ex. ett starttema, och göra något eget blir därmed flytande: Många befintliga teman har en liknande struktur och mycket av den inkluderade koden är kopierad och klistrad från något annat tema, WordPress.org:s egna sidor eller någon annan webbsida. Det är också vanligt att inkludera ramverk av olika slag, så som Bootstrap (https://github.com/twbs/bootstrap) eller Foundations (http://foundation.zurb.com/) för utseendet eller t.ex. UpThemes Framework (https://github.com/UpThemes/UpThemes-Framework/) för att på ett enkelt vis utöka Anpassaren (Customizer).

Fördelen med att skapa en egen grund är att man från början har koll på var olika HTML-element är placerade, och kan på ett mer aktivt sätt besluta om vad man vill utelämna alternativt inkludera.

2.2 Översikt över populära teman just nu (april 2015)

Jag har valt att ta en titt på de 20 mest populära teman just nu, vecka 17, för att få en uppfattning om hur de är byggda.

2.2.1 Avgränsningar

Jag har utgått ifrån den lista över populära gratisteman som finns på WordPress hemsidor (WordPress:o). Det är värt att nämna att jag inte vet på vilken basis denna lista, eller snarare rutnät, är uppgjord, men utgående från de markanta skillnaderna i totala antal nedladdningar mellan de teman jag har tittat på är det klart att det i varje fall inte är detta som avgör ett temas placering.

Varje år ger Automattic ut ett nytt tema som är namngivet efter det aktuella året. Eftersom dessa teman inkluderas i varje nedladdning av WordPress har jag valt att inte ta med dem i min översikt.

Det jag presenterar nedan är endast en snabb överblick, för att ge en beskrivning av vilka slags som teman är populära.

2.2.2 Observationer

Det som framkommer är att en majoritet (¾) av de teman jag tittade på var s.k. lite-version, vilket betyder att det finns en betalversion av samma tema som erbjuder utvidgade möjligheter till att skräddarsy webbsidans utseende via ett grafiskt användargränssnitt. Också det faktum att WordPress verkar vara ett populärt CMS för småföretags webbsidor[10] syns i många temans beskrivningar: Hälften av dem har ordet ”business” i sin beskrivning och 11 av 20 teman är anpassade för webbaffärstillägget WooCommerce. Jag kan tänka mig att företag är mer villiga att betala för ett tema än vad bloggare kunde tänkas vara och att temabyggare därför ser en möjlig inkomstkälla i att försöka vinkla sina teman till företag.

[10] Det här är min egen bedömning av läget.

En fjärdedel av de teman jag tittat på innehåller en kommentar i `style.css`-filens sidhuvud som nämner att temat baserar sig på starttemat Underscores. Ett par teman har använt ramverket Options för att utvidga anpassarpanelen och Bootstrap förekommer i 6 stycken teman, om än jag inte granskat till vilket syfte Bootstrap har inkluderats.

Det som förvånande mig var att det fanns ett tema, evolve, som hade flera brister i sin kod, bl.a. gick det att finna dessa rader i temats funktionsfil:

```
if ( get_stylesheet_directory() == get_template_directory() ) {
 define('EVOLVE_URL', get_template_directory() .
'/library/functions/');
 define('EVOLVE_DIRECTORY', get_template_directory_uri() .
'/library/functions/');
} else {
 define('EVOLVE_URL', get_template_directory() .
'/library/functions/');
 define('EVOLVE_DIRECTORY', get_template_directory_uri() .
'/library/functions/');
}
```

Indenteringen var också okonventionell. [11]

evolve hade fått kritik för att det lär ha en tendens att gå sönder vid uppdateringar, vilket pekar på att den som uppdaterat temat inte funderat på vilka konsekvenser olika ändringar i koden kan ha (WordPress:ä). Det finns också skäl att misstänka att den eller de som arbetar på temat uppdaterar temat ofta för att på så vis förhöja antalet nedladdningar. Det här föranledde mig att undra hur temat hade klarat av granskningen som WordPress gör på varje tema och varje ny version av ett tema, eftersom det verkar som att granskarna tar sin uppgift på allvar. [12] Det är värt att nämna att även andra teman saknade obligatoriska filer, samt där en del av indenteringen var ofullständig, men evolve särskilde sig.

---

[11] evolve v 3.2.7 går att titta på: https://themes.trac.wordpress.org/browser/evolve/3.2.7

[12] Se till exempel valbar Ticket på ett tema som håller på att granskas https://themes.trac.wordpress.org/query?status=reopened&status=reviewing&col=id&col=summary&col=owner&col=priority&col=resolution&col=time&col=changetime&col=reporter&report=8&desc=1&order=changetime

WordPress önskar också att temaskapare skall ta eventuella barnteman i beaktande och kolla om vissa funktioner redan existerar. Det här gäller främst för initieringsfunktionen (se kapitel 4.2.2). Fyra av de teman jag tittade på hade inte en if-sats som kollade om det redan fanns en initieringsfunktion. Därmed kommer eventuella barntemans förändringar att överskrivas av föräldertemat.

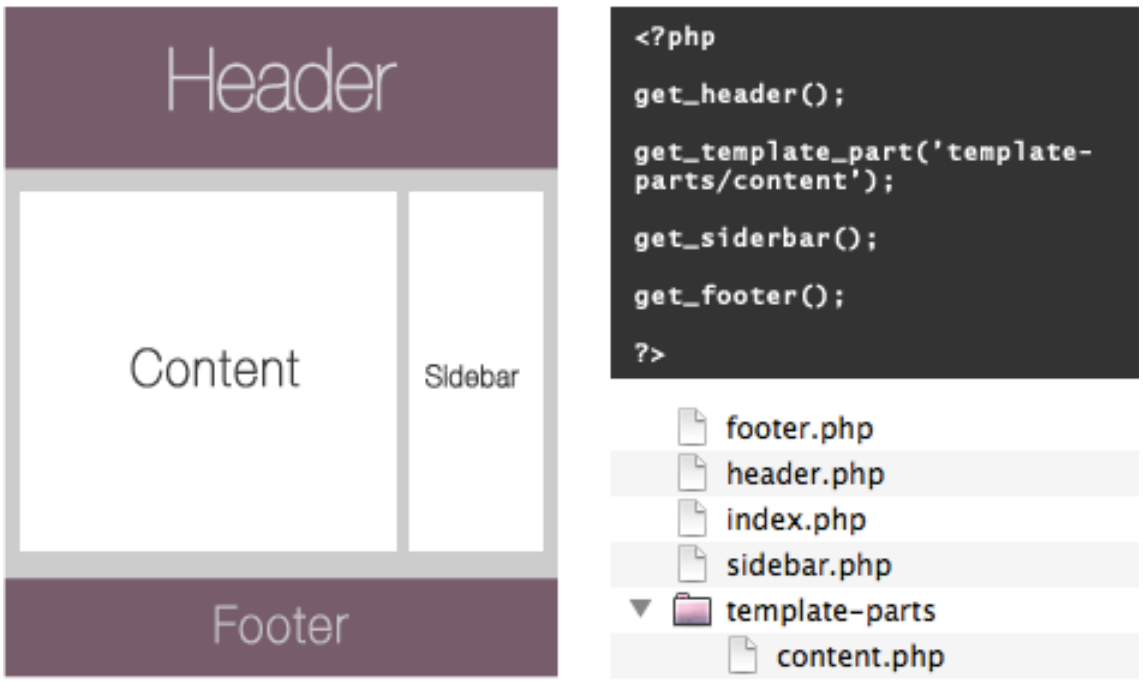
3 WORDPRESS-TEMANS STRUKTUR

3.1 Traditionell layout

Den traditionella layouten för en WordPress webbsida har ett sidhuvud, ett element som innehåller blogginläggen, en sidospalt (sidebar) som innehåller s.k. widgets (tillbehör)[13]. Därtill finns en navigeringsmeny, ett sökfält och en sidfot (footer). (de Valk, 2011)

Nedan följer ett simpelt exempel på hur layouten på en sida relaterar till dess sidmall (Page Template, se kapitel 3.3.2) och filstrukturen i ett tema:

[13] Det finns ingen svensk översättning för widgets. Till widgets hör arkiven, etikettmoln, etiketter, kategorierna. Många tillägg (Plugins) placerar också funktionalitet i widget-området.



Figur 2. Exempel på en sidmall (Page Template, se kapitel 3.3.2) vid namnet index.php och hur den relaterar till den färdiga sidan. All kod ovan är s.k. malltaggar (Template Tags, se kapitel 3.3.3) som letar fram mallfilernas (Template Files, se kapitel 3.3.2) innehåll.

3.2 Viktiga filer

Det finns ett antal obligatoriska filer som måste finnas med i teman, men två av dessa särskiljer sig, nämligen stilfilen `style.css` och funktionsfilen `functions.php`.

3.2.1 Stilfilen `style.css`

Stilfilen `style.css` är den fil som förutom att definiera temats utseende, eller i varje fall en del av det, innehåller den information om temat som syns i administrationspanelen samt i WordPress temakatalog, ifall temat har lagts till dit. Informationen är placerad i stilfilens sidhuvud.

WordPress klargör inte vilka taggar som är obligatoriska men nedan följer ett exempel på några som det kan löna sig att inkludera:

```
/*
Theme Name:           Temats namn
Author:               Författarens namn eller nickname
Author URI:           författarens webbsida
Description:          Text som handlar om och informerar om te-
                      mats ändamål
Version:              Versionnummer t.ex. 1.0
License:              GNU General Public License v2 or later
License URI:          http://www.gnu.org/licenses/gpl-2.0.html
*/
```

3.2.2 Funktionsfilen `functions.php`

Funktionsfilen `functions.php` behövs för att möjliggöra tilläggandet av funktionalitet till en WordPress webbsida. WordPress kräver att viss funktionalitet skall tilläggas, men ifall ett tema inte är ämnat för allmänt bruk kan filen utelämnas.

Den tillagda funktionaliteten är i bruk endast då temat är aktiverat och gäller endast för det tema vars funktionsfil en funktionalitet är inkluderad i. (WordPress:f)

För att utföra funktionella förändringar samt lägga till funktionalitet måste temautvecklaren använda sig av s.k. krokar (hooks). Krokarna delas in i två grupper:

1. filterkrokar (Filter Hooks)
2. händelsekrokar (Action Hooks)

Man skapar alltså en funktion som man sedan krokar till en filter- eller händelsekrok med funktionen `add_filter()` respektive `add_action()`[14], se exemplet i kapitel 2.1.1. Typen av krok definierar när funktionen skall köras ([Griffiths], [2010])

På många sätt fungerar funktionsfilen som ett tillägg (Plugin) och därför är det viktigt att komma ihåg att ifall den funktionalitet man lägger till borde vara oberoende av utseendet så bör man skapa ett tillägg istället. (WordPress:z)

[14] Även om dessa har olika uppgifter – filter filtrerar input och händelse utför en åtgärd, så är de grundläggande `add_action`- och `add_filter`-funktionerna desamma och är utbytbara sinsemellan. (WordPress:c) ([Griffiths, [2010])

3.3 Central terminologi

WordPress har en egen terminologi[15] för hur den delar in de .php-filer som genererar webbsidans HTML-dokument. Dessa är:

1. Mallfiler (Template Files)
2. Sidmallar (Page Temaplates)[16]

I mallfilerna och sidmallarna kan det förekomma kod i form av

1. Malltaggar (Template Tags)
2. Villkorstaggar (Conditional Tags)
3. Slingan (The Loop)

samt vanlig HTML- och PHP-kod.

3.3.1 Mallfiler (Template Files)

En mallfil (Template File), eller mall (Template), är en .php-fil som innehåller PHP- och eventuellt HTML-kod. En mallfil innehåller antingen samtliga delmoment som behövs för att generera ett giltigt HTML-dokument eller endast ett parti av ett giltigt HTML-dokument.

De mallfiler, som inte är sidmallar (Page Template, se kapitel 3.3.2), bidrar till att göra temats olika beståndsdelar mer överskådliga. Det är bra att dela upp innehållet i mindre

[15] Härmed inte sagt att den terminologi som WordPress använder sig av är unik för WordPress. Användandet av terminologi som innehåller ordet ”mall”, d.v.s. ”template”, är vanligt bland teknologier som fungerar på serversidan, d.v.s. system som bygger på bl.a. PHP, Django och/eller Ruby (Kitamura, 2014). Överlag är bruket av ordet ”template” vanligt inom datorbaserad teknik, se t.ex. Microsoft Word och Adobe Illustrator.

[16] Det finns också arkivmallar, sökmallar etc., men de liknar sidmallar till koden, så jag anser att de också kan räknas som sidmallar.

delar, eftersom man på så vis kan undvika upprepning och temat blir mindre. (WordPress:v)

3.3.2 Sidmallar (Page Templates)

En sidmall (Page Template) visar en WordPress-webbsidas dynamiska innehåll på en sida, alternativt skulle man kunna säga att en sidmall genererar ett giltigt HTML-dokument. (WordPress:s).[17]

I sidmallen används (inkluderande) malltaggar (Template Tags, se kapitel 3.3.3) som letar fram innehållet i de mallfiler (Template Files) som åkallas, samt eventuellt annan PHP- och HTML-kod för att m.h.a. dessa generera ett fullständigt HTML-dokument.

Det finns två typer av sidmallar:

1. Sådana som kan användas globalt
2. Sådana som används i ett sammanhang s.k. engångsmallar

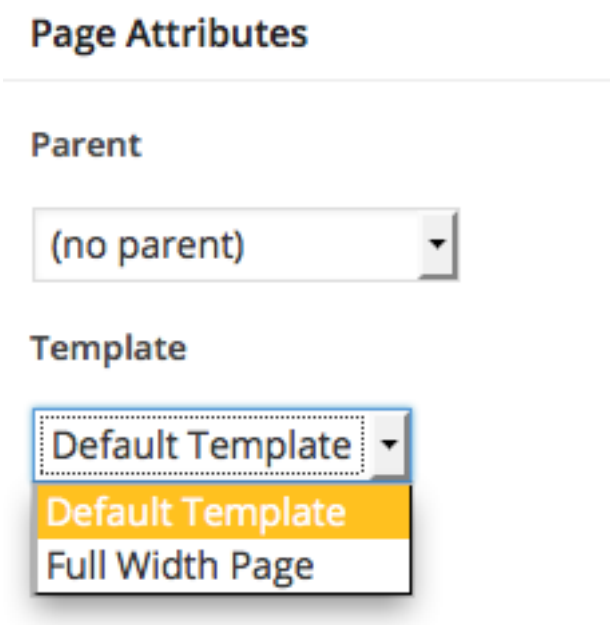
3.3.2.1 Globala sidmallar

Globala sidmallar får namnges enligt skaparens önskemål, men namnet får inte börja med ”page-”, eftersom WordPress automatiskt tolkar dessa som engångsmallar (single use). De får inte heller namnges med ett filnamn som WordPress har reserverat. Globala sidmallar kan väljas via administrationspanelen och måste därför ha ett sidhuvud där deras mallnamn (Template Name) definieras. De globala sidmallarna finns alltså till för att temaanvändaren skall ges en möjlighet att påverka utseendet på sin webbsida, t.ex. genom att kunna välja bort sidospalterna (Sidebars).

Man skall gärna lägga globalt applicerbara sidmallar i en mapp vid namnet `page-templates`. I andra fall skall de placeras i temats huvudmapp. (WordPress:s)

[17] Det verkar som att termen sidmall har förtydligats i.o.m. Handboken - i Codexen presenteras sidmallar på ett mindre framträdande vis. (jmf. WordPress:i med WordPress:v)

```
/**
 * Template Name: Full Width Page
 *
 * @package WordPress
 * @subpackage Temats_Namn
 * @since Temats Namn 1.0
 */
```



Figur 3. Exempel på hur sidhuvudet i en sidmall relaterar till hur man kan välja mall för en sida.

3.3.2.2 Engångsmallar

Engångsmallar eller specialiserade mallar används för en unik sida. För sidor gäller namn av typen `page-{slug}.php`[18] eller `page-{sidans ID-nummer}.php`.

Det går också att skapa engångsmallar för bl.a. kategorier, etiketter och författare enligt samma modell som för sidor. Istället för `page` använder man då `category`, `tag` respektive `author`. Dessa kallas inte för sidmallar, utan istället för kategori-, etiketts-

[18] Ordet *slug* har ingen direkt svensk motsvarighet, och jag vågar mig därmed inte på att komma med något eget ord. Med *slug* avses en del av en URL som definierar en för människor läsbar URL. Till exempel i http://webbsida.com/om-mig är om-mig en *slug*.

och författarmallar och även de bör innehålla alla delmoment som behövs för att generera en fullständig webbsida.

Engångsmallar får inte placeras i underliggande mappar och ifall ett barntema används måste engångsmallarna vara placerade i barntemats mapp. Överlag är engångsmallar avsedda för att skräddarsy ett tema och de bör inte inkluderas i teman som laddas ned via WordPress.org.

3.3.2.3 Mallhierarki (Template Hierarchy)

WordPress har en **mallhierarki** (Template Hierarchy)[19] som är till för att styra vilken sidmall som skall användas i den aktuella sökningen. En sökning (Query) sker då någon eller något besöker webbsidan samt då en länk aktiveras. Om ingen passande sidmall hittas faller WordPress alltid tillbaka på `index.php`-filen. (WordPress:w)

3.3.3 Malltaggar (Template Tags)

Mallfiler (Template Files) och sidmallar (Page Templates) består, som nämnts, av HTML- och PHP-kod. En del av PHP-koden är s.k. malltaggar (Template Tags). Med malltaggar avses WordPress egna funktioner och de används för att utföra åtgärder, leta fram information ur databasen eller leta efter mallfiler. (WordPress:h)

Den form av malltaggar som letar fram mallfiler kallas för inkluderande taggar (Include Tags). Ett exempel på en inkluderande tagg är `get_header()`. Denna används för att leta fram `header.php`-filen, som måste vara placerad i temats egen mapp och inte i en undermapp. (WordPress:g) Det verkar dock som att WordPress håller på att gå ifrån denna indelning och i Handboken omnämns inte inkluderande taggar skiljt (WordPress:x).

[19] Bilden över mallhierarkin är alltför stor för att inkludera som bilaga, men den finns tillgänlig på webben: http://wphierarchy.com/ och https://codex.wordpress.org/File:wp-template-hierarchy.jpg

Man kan ofta påverka resultatet en malltagg ger genom att skicka in en sträng (string), eller strängar alternativt en eller flera uppställningar (array) som argument. För att veta vilka argument som är gångbara måste man kolla upp den ifrågavarande malltaggens API. (WordPress:b)

3.3.4 Villkorstaggar (Conditional Tags)

Villkorstaggar (Conditional Tags) nämns i anknytning till malltaggar, men är inte malltaggar. De börjar på `is_` och används i if-satser så som villkor. Som exempel kan nämnas `is_front_page()` som kollar om den ifrågavarande sidan är första-sidan. Alla villkorstaggar ger svaret `true` eller `false`. (WordPress:q)

3.3.5 Slingan (The Loop)

För att kunna visa mer än endast ett inlägg (Post) behöver temat inkludera den s.k. Slingan (The Loop). Slingan består av malltaggar som letar fram inlägg eller en sida i enlighet med den aktuella sökningen (Query):

```
<?php
if ( have_posts() ) :
    while ( have_posts() ) : the_post();
        /*
        Innanför denna slinga finns de villkorssatser,
        malltaggar och den HTML-kod som behövs för
        att skapa den önskade utskriften.
        */
    endwhile;
endif;
?>
```

Det finns en del malltaggar (Template Tags) som endast fungerar innanför Slingan. (WordPress:y)

4 PRAKTISKT ARBETE

Som hjälpmedel för uppbyggandet av grunden för mitt tema har jag använt mig av Automattics egna teman, d.v.s. jag har studerat de färdiga temans kod och använt mig av Handboken, samt delar av Codexen för att förstå mig på hur de olika malltaggarna och krokarna skall användas.

Det jag kommer beskriva nedan är inte en fullständig beskrivning av hur man bygger ett tema – min strävan är att förtydliga det jag gick igenom i föregående kapitel, samt ge den kod och de beskrivningar som behövs så att man bättre skall kunna förstå de olika partierna i ett WordPress-tema. I mina exempel saknas därmed några obligatoriska malltaggar och jag nämner inte heller exakt vilka mallfiler som måste inkluderas. För en fullständig lista över obligatoriska filer och malltaggar måste man studera WordPress.org:s. Det finns därtill ett tillägg som bör användas för att kolla igenom ett tema före det laddas upp för granskning.

4.1 Minimikrav

För att ett tema skall kunna identifieras av WordPress-installationen behöver det, som nämnts, en egen mapp som innehåller en stilmallsfil med namnet `style.css`, samt en `index.php`-fil.

För att skapa min första `index.php`-sida arbetade jag utifrån det exempel som finns i Handboken (WordPress:å). Jag ville bygga en simpel grund utifrån vilken det vore enkelt att utöka innehållet och dela upp i delar. Nedan följer ett exempel på en minimal `index.php`-sida, som klarar av att visa en meny, som innehåller länkar till alla skapade sidor, samtliga inlägg tillsammans med paginering alternativt en enskild sida.

Nästan all PHP-kod i mitt exempel utgörs av malltaggar (Template Tags) och villkorstaggar (Conditional Tags). Det finns därtill två händelsekrokar nämligen `wp_head()` och `wp_footer()`.

I stilfilen inkluderade jag till att börja med endast ett sidhuvud med temats namn.

```
<!DOCTYPE html>
<html <?php language_attributes(); ?> >
    <head>
        <meta charset="<?php bloginfo( 'charset' ); ?>">
        <link rel="stylesheet" href="<?php echo esc_url(
get_stylesheet_uri() ); ?>" type="text/css" />
        <?php wp_head(); ?>
    </head>
    <body>
        <header>
           <h1><a href="<?php echo esc_url( home_url() ); ?>">
           <?php bloginfo( 'name' ); ?></a></h1>
           <h2><?php bloginfo( 'description' ); ?></h2>
           <?php wp_page_menu(); ?>
        </header>

        <main>
            <?php if ( have_posts() ) :
                 while ( have_posts() ) : the_post(); ?>

                    <div <?php post_class(); ?> >

                        <h3><?php the_title(); ?></h3>
                        <?php the_content(); ?>

                    </div>

                <?php endwhile; ?>

                <?php echo paginate_links( ); ?>

            <?php else : ?>

                <p>No posts found. :(</p>

            <?php endif; ?>
        </main>

        <footer>
            <?php wp_footer(); ?>
        </footer>
    </body>
</html>
```

Figur 4. Enkel index.php-fil

4.2 Funktionsfilen functions.php

Det finns funktionalitet som måste "registreras" via funktionsfilen. Därutöver har WordPress.org en lista på funktionalitet som det rekommenderas att skall inkluderas i teman. WordPress rekommenderar också att samtliga funktioners namn skall börja med temats slug, d.v.s. en form av temats namn som varken innehåller versaler eller tecken som inte ingår i det engelska alfabetet, för att det inte skall kunna ske någon sammanblandning mellan temats och eventuella tilläggs (Plugins) funktioner.

Då det i mina exempel förekommer `'temats_namn'` så som argument i en funktion och inte så som namnet på en funktion är det frågan om temats s.k. textdomän. Textdomänen behövs för att temat skall kunna översättas.

4.2.1 Testa om funktionen redan finns

De flesta funktioner som man skapar i funktionsfilen bör omges med en if-sats som kollar om den ifrågavarande funktionen inte redan finns. Detta behövs för att ett eventuellt barntema skall kunna ändra på den inkluderade funktionaliteten. Därmed är prövningen relevant för de funktioner som en skapare till ett barntema skulle tänkas vilja ändra på.

```
if ( ! function_exists( 'temats_namn_funktionens_namn' ) ) :
    function temats_namn_funktionens_namn() {
        //diverse
    }
endif;
```

4.2.2 Initieringsfunktionen

En majoritet av den funktionalitet som borde inkluderas i ett tema krokas till händelsekroken (Action Hook) `after_setup_theme`, innanför en s.k. initieringsfunktion.

Initieringsfunktionen kan döpas till vad som helst, men för tydlighetens skull brukar namnet innehålla antingen ordet ”setup” eller ”init”. Nedan följer ett exempel som både innehåller sådan funktionalitet som är obligatorisk och sådan som rekommenderas. (WordPress:z)

```
if ( ! function_exists( 'temats_namn_setup' ) ) :

    function temats_namn_setup() {

        // Möjliggör RSS-flöden. Obligatorisk
        add_theme_support( 'automatic-feed-links' );

        /* Lägger till menyer. Möjliggör det för temautvecklaren
         * att avgöra var olika menyer är placerade.
         * Rekommenderas
        */
        register_nav_menus( array (
            'header-nav'  => __( 'Header Menu', 'temats_namn' ),
        ));

        //Möjliggör Post Thumbnails. Rekommenderas
        add_theme_support( 'post-thumbnails' );

        /* Möjliggör användandet av inläggsformat (Post Formats)
         * (kräver mallfiler). Rekommenderas
        */
        add_theme_support( 'post-formats',  array ( 'aside',
'gallery', 'quote', 'video', ) );

        // Möjliggör översättningar av temat
        load_theme_textdomain( 'temats_namn',
get_template_directory() . '/languages' );

         // Lägger till <title> till wp_head()
        add_theme_support( 'title-tag' );
    }
endif;

add_action( 'after_setup_theme', 'temats_namn_setup' );
```

4.2.3 Sidospalt (Sidebar)

Att möjliggöra användningen av sidospalter (sidebars) som innehåller s.k. widgets, d.v.s. arkiv, etikettmoln o.s.v., är obligatoriskt. Som temautvecklare kan man själv bestämma hur många sidospalter man vill lägga till i sitt tema.

Registrerandet av sidospalter görs i funktionsfilen på följande vis:

```
if ( ! function_exists( 'temats_namn_add_sidebar' ) ) :

    function temats_namn_add_sidebar() {

        /*
         * Temat innehåller endast en sidospalt.
        */
        register_sidebar( array(
            'name'           => __( 'Sidebar', 'temats_namn' ),
            'id'             => 'the-sidebar',
            'description'    => 'Widgetarna är placerade till
höger',
        ) );

    }
endif;

add_action( 'widgets_init', 'temats_namn_add_sidebar' );
```

I likhet med initieringsfunktionen behövs det en funktion som sedan krokas till en händelsekrok, i det här fallet `widgets_init`.

För egen del inkluderade jag endast en sidospalt. Det finns en funktion, `register_sidebars( $number, $args )`, för att inkludera flera sidospalter på en gång, men det rekommenderas ändå att man skall använda upprepade `register_sidebar()`, eftersom man via den kan ge varje sidospalt ett eget unikt namn som användaren kan se. (WordPress:d)

4.2.4 JavaScript- och stilmallsfiler

För att inkludera andra stilmallsfiler, förutom `style.css`, och JavaScript-filer skall man använda sig av `wp_enqueue_style()` respektive `wp_enqueue_script()`. Dessa krokas sedan till händelsekroken `wp_enqueue_scripts()`. Då det kommer till JavaScript lönar det sig att kolla vilka bibliotek som finns med i WordPress-installationen, så att man inte inkluderar onödiga filer i sitt tema (WordPress:e).

```
if ( ! function_exists( 'temats_namn_scripts' ) ) :

    function temats_namn_scripts() {

        wp_enqueue_style( 'style', get_stylesheet_uri() );
        wp_enqueue_script( 'own-script',
get_template_directory_uri() . '/js/scriptname.js', array(), '',
true );

}

endif;

add_action( 'wp_enqueue_scripts', 'temats_namn_scripts' );
```

I exemplet ovan har stilfilen lagts till med `wp_enqueue_style()` istället för att inkluderas direkt i `<head>`-elementet så som i `index.php`-exemplet.

I mitt eget tema använde jag mig av en kodsnutt från Twenty Fifteen som genererar en sökväg för Google Fonts i enlighet med användarens språkinställningar. Därtill lade jag till en egen fil med JavaScript-kod. Jag valde också att inkludera temats stilfil på samma sätt som i exemplet i det här kapitlet.

4.3 Utvidgad filstruktur

Efter att ha satt ihop majoriteten av funktionsfilen var det dags att börja dela upp `index.php`-filen i mallfiler (Template Files) och skapa olika sidmallar (Page Templates).

Allt som allt måste ett tema, som kan inkluderas i temakatalogen, innehålla 10 stycken `.php`-filer. En av dessa filer är funktionsfilen och 5 av filerna kan struktureras som sidmallar. Därtill behövs stilfilen `style.css`. (WordPress:t)

4.3.1 Mallfiler

4.3.1.1 header-, footer- och content.php

Det är lättast att börja skapa de olika delarna `header`-, `footer`- och `content.php` genom att dela upp innehållet i `index.php`-filen. `content.php` är inte en obligatorisk fil, men de andra delarna är det och därmed är det vanligt att lägga mittpartiet i en egen mallfil (Template File).

På WordPress.org:s sidor nämns den huvudsakliga navigeringsmenyn i anknytning till `header.php`, vilket stämmer överens med majoriteten av de teman jag har tittat på. Därmed brukar det HTML-element som innehåller största delen av det som Slingan (The Loop) skriver ut börja i `header.php` och avslutas i `footer.php`.

Inkluderandet av händelsekrokarna (Action Hooks) `wp_head()` och `wp_footer()`, som finns med i mitt exempel på en simpel `index.php`-fil, är obligatoriskt.

4.3.1.2 Sidospalter

Det är obligatoriskt att inkludera en fil vid namnet `sidebar.php` i sitt tema. Denna brukar innehålla koden för temats huvudsakliga sidospalt. För att WordPress skall kunna skriva ut sidospaltens innehåll måste malltaggen (Template Tag) `dynamic_sidebar( 'sidospaltens_id')` användas.

```
<?php if ( is_active_sidebar( 'the-sidebar' ) ) : ?>
    <div id="sidebar" class="sidebar" role="complementary">
        <?php dynamic_sidebar( 'the-sidebar' ); ?>
    </div>
<?php endif; ?>
```

I exemplet ovan finns det inget `<div>` -element om inte sidospalten har aktiverats. Sidspalten aktiveras när användaren lägger in tillbehör (widgets) i den via administrationspanelen.

För att sedan i en sidmall komma åt innehållet i `sidebar.php` används malltaggen `get_sidebar()` (se figur 2). Andra sidospalters `.php`-filer namnges i stil med `sidebar-other.php` och åberopas med `get_sidebar( 'other' )`.

Det är också värt att nämna att sidospalter på inga vis måste vara placerade i sidfält - de kan vara placerade var som helst på sidan, t.ex. i sidfoten. (WordPress:u)

4.3.1.3 Kommentarer

Liksom `sidebar.php` är det obligatoriskt att inkludera en `comments.php`-fil i sitt tema.

```
<?php if ( ! post_password_required() ) : ?>

    <div id="comments" class="comments-area" >

        <?php if ( have_comments() ) : ?>

            <h2 class="comments-title" >
                <?php printf( _nx( 'En kommentar angående %2$s',
'%1$s kommentarer angående %2$s', get_comments_number(),
'comments title', 'temats_namn' ), get_comments_number(),
get_the_title() ); ?>
            </h2>

            <ol class="comment-list">
                <?php
                    wp_list_comments( array(
                        'style'       => 'ol',
                        'short_ping'  => true,
                        'avatar_size' => 74,
                    ) );
                ?>
            </ol>

        <?php endif; /* have_comments() */ ?>

        <?php comment_form(); ?>

    </div> <!-- #comments -->

<?php endif; ?>
```

För att överhuvudtaget kunna se kommentarerna för ett inlägg eller en sida får denna inte vara försatt med lösenord. Ifall det finns kommentarer skriver WordPress i detta exempel ut antingen “En kommentar angående [inläggets namn]” eller “[nummer] kommentarer angående [inläggets namn]” beroende på vad `get_comments_number()` ger för svar. Funktionen `_nx( $single, $plural, $number, $context, $domain )` är en av WordPress översättningsfunktioner.

`wp_comments_list()` skriver ut själva kommentaren med en del HTML-element. `comment_form()` skriver ut kommentarfältet. (WordPress:p)

WordPress kommer automatiskt att dölja kommentarerna i blogg-läge, men visa dem om man tittar på ett inlägg eller en sida. För att göra kommentarer synliga också i bloggläge kan man lägga in

```
$withcomments = true;
```

i `functions.php`-filen.

Malltaggen `comments_template()` används för att skriva ut `comments.php`:s innehåll. Den skall placeras innanför Slingan (The Loop).

4.3.2 Sidmallar och liknande

Förutom default-sidmallen `index.php` måste temat innehålla sidmallarna `single.php`, för enskilda inlägg och `page.php` för sidor. Dessa två kan man till att börja med skapa utgående från `index.php`-sidan.

Därtill är arkivmallen `archive.php` obligatorisk. Den fungerar som defaultsida för alla arkivsökningar på författare, kategori, etikett, datum samt på anpassade inläggstyper och taxonomitermer. Arkivsidan kan liksom sidmallarna skapas utgående från index.php.

I.o.m. WordPress 4.1 kom två nya malltaggar (Template Tags) som är ämnade för att användas vid framställningen av arkiv: `the_archive_title()` och `the_archive_description()`. Dessa skall läggas före Slingan (The Loop).

Sökmallen `search.php` är också obligatorisk och liksom `archive.php` liknar den de andra sidmallarna förutom att titeln skall vara sökningen. För detta finns funktionen `get_search_query()`:

```
<header class="search-header">
    <h1 class="search-title">
        <?php printf( __( 'Sökresultat för: %s', 'temats_namn'),
get_search_query() ); ?>
    </h1>
</header>
```

__() är precis som _nx() en översättningsfunktion. Sökresultatets titel placeras ovanför Slingan.

4.4 Utvidgad funktionalitet

4.4.1 Navigeringsmenyer

WordPress har två malltaggar (Template Tags) som kan användas för att visa menyer.

wp_nav_menu() måste, som nämnts i kapitel 4.2.2, definieras i functions.php med register_nav_menus() innanför initieringsfunktionen.
wp_nav_menu('menyns_id') skriver ut menyer som användaren har definierat, d.v.s. användaren har valt vilka sidor som skall länkas i den ifrågavarande menyn samt kryssat i vilken meny det är frågan om. Om inga menyer har registrerats finns det inte heller någon meny att kryssa i på administrationspanelens menyvy.

wp_page_menu() är WordPress fallbackfunktion för de gånger wp_nav_menu() åkallas utan att användaren har definierat någon meny, d.v.s. angett vilka sidor som skall länkas i den specifika menyn. wp_page_menu() går också att använda som sådan, men den erbjuder inga möjligheter för användaren att bestämma vilka specifika sidor som skall synas i dess utskrift.

I mitt tema skapade jag en egen fallbackfunktion för wp_nav_menu(). Jag måste göra detta eftersom jag ville att mina menyer skall omges av ett <nav>-element och wp_page_menu() erbjuder inte, till skillnad från wp_nav_menu(), rätten att påverka vilket element som skall fungera som hållare (container).

4.4.2 Inkluderandet av anpassningsalternativ

4.4.2.1 Anpassat sidhuvud (Custom Header)

Att ge användaren möjligheten att lägga upp en egen bild i sidhuvudet på sin WordPress-sida är antagligen det vanligaste anpassningsalternativet som temautvecklare lägger till i sina teman. Funktionaliteten läggs till i initialiseringsfunktionen (se kapitel 4.2.2) med funktionen `add_theme_support('custom-header', $args)`.

Som andra argument ($args) går det att lägga en uppställning (array) i vilken det bl.a. går att definiera en default-bild för sidhuvudet. För att sedan kunna visa det anpassade sidhuvudet måste man antingen lägga till en bild med <img> -elementet eller en bakgrundsbild med CSS.

Att lägga in ett bildelement är det enklare alternativet, då bildtaggen bara måste läggas in i sidhuvudsfilen:

```
<header id="page-header">
    <img src="<?php header_image(); ?>"/>
</header>
```

4.4.2.2 Anpassad bakgrund

Det finns också ett `'custom-background'`-argument som läggs till på samma vis som det anpassade sidhuvudet, dvs genom att lägga till `add_theme_support('custom-background', $args)` i initialiseringsfunktionen. Den anpassade bakgrunden fungerar inte på samma vis som sidhuvudet: WordPress vill varken visa defaultbild eller bakgrundsfärg utan att användaren har definierat dem. Det går att ändra på detta t.ex. genom att skapa en egen fallbackfunktion, men eftersom jag i mitt tema bara vill ge användaren möjligheten att påverka bakgrundsfärgen beslöt jag mig för att lägga till ett eget anpassningsalternativ istället.

4.4.2.3 Ett eget anpassningsverktyg

För att lägga till en egen Anpassare (Customizer) måste man skapa tre funktioner och en JavaScript-fil:

1. En funktion som registrerar själva anpassaren och skapar ett användargränssnitt för den
2. En funktion som lägger till CSS i sidhuvudet
3. En JavaScript-fil som innehåller den kod som behövs för att kunna förhandsvisa det valda alternativet.
4. En funktion som lägger till JavaScript-filen till sidfoten

Samtlig PHP-kod som presenteras nedan kan placeras i `functions.php`-filen, men det är också vanligt att skapa en egen fil för Anpassaren.

Steg 1. En anpassare består av en sektion, en kontroll och inställningar. (McFarlin, 2013)

Figur 5. En bakgrundsanpassare

För att skapa en likadan anpassare som den som presenteras i bilden ovanför (Figur 5) behövs alltså följande kod:

```
function temats_namn_customizer( $wp_customize ) {

    $wp_customize->add_setting( 'background_color' , array(
       'default'     => '#565584',
       'transport'   => 'postMessage',
    ) );

    $wp_customize->add_control( new WP_Customize_Color_Control(
        $wp_customize,
        'temats_namn_background_color',
        array(
            'label'      => __( 'Background Color',
'temats_namn' ),
            'section'    => 'colors',
            'settings'   => 'background_color',
    ) ) );
}

add_action( 'customize_register', 'temats_namn_customizer' );
```

I kodsnutten ovan genereras en inställning för `'background_color'`. Inställningen läggs sedan till i den färgkontroll (`WP_Customize_Color_Control`) som skapas. Sektionen (`'section'`) anger var färgkontrollen skall vara placerad. Jag har valt placera den i Colors (Färger), som är en färdigt definierad sektion.

Steg 2: Skapa en funktion som genererar en stiltagg:

```
function temats_namn_add_style_tag() {
    ?>
         <style type="text/css">
             body {
                 background-color: #<?php echo
get_theme_mod('background_color', '565584'); ?>;
             }
         </style>
    <?php
}

add_action( 'wp_head', 'temats_namn_add_style_tag' );
```

`get_theme_mod()` kollar om det finns ett värde för `'background_color'`. Om ingen färg har definierats väljs den andra parametern d.v.s. 565584.

Steg 3. Skapa en JavaScript-fil som kan förhandsvisa det som valts med färgkontrollen m.h.a. `wp.customize`-objektet:

```
wp.customize( 'background_color', function( value ) {
      value.bind( function( newval ) {
            document.body.style.backgroundColor = newval;
      } );
} );
```

Steg 4. JavaScript-filen skall laddas med `wp_enqueue_script()` och krokas till händelsekroken `customize_preview_init()` (se t.ex. kapitel 4.2.4). Det finns alltså en krok som bara finns till för att kunna förhandsvisa det som görs med Anpassaren. (WordPress:k)

5 DISKUSSION

I.o.m. den grundliga genomgång jag gjort av hur teman är uppbyggda har jag fått en betydligt bättre förståelse för hur WordPress fungerar. Innan jag hade gått igenom WordPress terminologi för teman var det svårt för mig att förstå hur de olika (mall)filerna relaterade till varandra och funktionsfilens innehåll sade mig inget. Nu när jag tittar på ett tema förstår jag hur det är uppbyggt och de olika filernas funktion.

Det har varit av betydelse för mig att jag har tittat på andra teman, men det som i många fall försvårat min förståelse av dem är att de ofta innehåller en hel del extra funktionalitet, som istället kunde inkluderas med olika tillägg. Det finns också mycket som talar för att de mest populära teman inte är särskilt representativa för vad människor laddar ned – de teman jag tittat på var skapade av företag och riktade sig till företag. De uppdaterades också rätt så ofta vilket bidragit till förhöjda nedladdningsantal – i grafen över nedladdningar för ett tema går det att se en korrelation mellan tidpunkten för en uppdatering och en plötslig uppgång i antalet nedladdningar.

Min genomgång av teman gav mig också en förståelse för hur teman granskas och bedöms. Temagranskarna (Theme Review Team) var noggranna med att vissa malltaggar, krokar och övriga funktioner skulle användas samt att rätt rättigheter gällde för material, ramverk och bibliotek som inkluderas i teman. Jag fann ändå ingen granskare som kommenterat kvaliteten på någons kod, varken för CSS eller PHP, fastän det finns riktlinjer för dessa. Temagranskarna verkar inte heller ha några befogenheter att neka att uppdatera ett tema i situationer då skillnaden mellan dess aktuella version och den nyare versionen är obetydlig.

WordPress har också negativa sidor, som både berör dess API och den terminologi som används. Då det kommer till det senare fäste jag mig framförallt vid termen sidmall (Page Template). Länge trodde jag att alla .php-filer som genererade ett fullständigt HTML-dokument kunde kallas för sidmallar, men sedan gick det upp för mig att de mallfiler som är ämnade för arkiv, sökresultat, etiketter, inlägg och taxonomi istället kallas för arkivmall, sökmall etc. Det här upplever jag som en onödig uppdelning, eftersom dessa inte skiljer sig på något avgörande sätt från sidmallar. Då det kommer till WordPress egna funktioner, som också delas in i många grupper (malltaggar (Template Tags), villkorstaggar (Conditional Tags), filter- och händelsekrokar (Filter and Action Hooks)) upplever jag fortfarande förvirring angående hur man t.ex. i vissa fall kan skilja på en krok och en malltagg.

Vad gäller hur WordPress funktioner fungerar har jag endast bekantat mig med en bråkdel. I min diskussion om hur `wp_nav_menu()` skiljer sig från det äldre `wp_page_menu()` ville jag ta fasta på att den senare inte har uppdaterats på fler år och att den därmed inte passar så bra som fallbackfunktion för `wp_nav_menu()`. Jag tycker också att när det kommer till de inkluderande malltaggarnas namn kunde man ha strävat efter en större enhetlighet: `get_header()` är rätt så självklar, men varför har motsvarande malltagg för kommentarer namngetts `comments_template()`?

Eftersom WordPress har blivit så populärt betyder det att varje uppdatering måste ta äldre versioner i beaktande, vilket antagligen gör det svårt att göra större förändringar i den befintliga koden. Då det kommer till teman har WordPress.org dock gått in för att inte understöda äldre versioner av WordPress och detsamma gäller säkert för tillägg.

Det här ser jag som ett bra beslut, eftersom det gör utvecklarnas jobb lättare då de inte behöver besluta om bakåtgående kompatibilitet. Det borde också bidra till att användare uppdaterar sina installationer oftare.

Genom min studie har jag ändå framförallt insett vilka möjligheter WordPress för med sig – som att man i barnteman kan skapa egna sidmallar, alternativt strukturera om en existerande mallfil. Det går alltså att skapa ett helt eget utseende som skiljer sig från originalet, men samtidigt hela tiden har någonting som fungerar.

Jag har också insett hur viktigt det är att gå igenom grunderna – då man behärskar dem verkar det mesta ganska enkelt. Nästa steg för mig kommer vara att bekanta mig med hur man bygger tillägg, så att jag skall få en bättre förståelse för det som nu fortfarande blev lite oklart för mig, nämligen krokarna.

LITTERATUR

Chandler, Jeff. (4 oktober 2008). *Ian Stewart on Child Themes, Part 1*. [forumsvar]. [Besökt 24 mars 2015]. Tillgänglig: <http://weblogtoolscollection.com/archives/2008/10/04/ian-stewart-on-child-themes-part-1/>

Davis, Jeff. (10 januari 2011). *Php: what's the difference between while…endwhile; and while*. [forum]. [Besökt 8 april 2015]. Tillgänglig: <http://stackoverflow.com/questions/4650971/php-whats-the-difference-between-while-endwhile-and-while-stuff-here>

Eastaugh, Benedict. (13 januari 2009). *Extralogical*. [blogg]. [Besökt 24 mars 2015]. Tillgänglig: <http://archive.extralogical.net/2008/08/theme-inheritance/>

[Griffiths, Everett]. ([2010]). *Anatomy of a WordPress Plugin*. [artikel]. [Besökt 29 april 2015]. Tillgänglig: <https://www.packtpub.com/books/content/anatomy-wordpress-plugin>

Kallinki, Antti and Pasanen Panu. (2013). *LAPSITEEMAN TOTEUTTAMINEN WORDPRESS-SIVUSTOLLE*. Oulu: Tietojenkäsittelyn koulutusohjelma. Oulun seudun ammattikorkeakoulu.

Kitamura, Eiji. (6 oktober 2014). *Introduction the the template elements*. [webbsida]. [Besökt 9 april 2015]. Tillgänglig: <http://webcomponents.org/articles/introduction-to-template-element>

McCollin, Rachel and Tessa Blakeley Silver. (2013). *WordPress Theme Development Beginner's Guide*. Olton: Packt Publishing.

McFarlin, Tom. (1 oktober 2013). *Theme Development. A Guide to the WordPress Theme Customizer: Sections, Settings, and Controls*. [webbsida]. [Besökt 20 maj 2015]. Tillgänglig: <http://code.tutsplus.com/tutorials/a-guide-to-the-wordpress-theme-customizer-sections-settings-and-controls--wp-33137>

Myllykangas, Mari. (2014). *WORDPRESS-TEEMAN TOTEUTTAMINEN*. Oulu: Tietojenkäsittelyn koulutusohjelma. Oulun ammattikorkeakoulu.

Mäkiniemi, Mikko. (2014). *Verkkosivujen suunnittelu WordPress-järjestelmällä*. Seinäjoki: Tekniikan yksikkö. Tietotekniikan koulutusohjelma. Seinäjoen ammattikorkeakoulu.

Poranen, Aapeli. (2015). *OHJELMOINTI WORDPRESS-YMPÄRISTÖSSÄ. Teemat, lisäosat ja multisite*. Mikkeli: Mikkelin ammattikorkeakoulu. Tietojenkäsittelyn koulutusohjelma.

Rooney, Zoe. (24 juli 2013). *Q & A: What does "custom" really mean?* [webbsida]. [Besökt 10 april 2015]. Tillgänglig: <http://zoerooney.com/blog/business/what-does-custom-really-mean/>

Sabin-Wilson, Lisa. (2013). *WordPress Web Design for Dummies*. Somerset, NJ, USA: John Wiley & Sons.

W3Techs. ([2015]). *Usage of content management systems for websites*. [webbsida]. [Besökt 23 april 2015]. Tillgänglig: <http://w3techs.com/technologies/overview/content_management/all>

de Valk, Joost. (10 januari 2011). *The antamoy of a WordPress theme*. [blogg]. [Besökt 4 mars 2015]. Tillgänglig: <https://yoast.com/wordpress-theme-anatomy>

Wappalyzer. ([2015]). *CMS*. [webbsida]. [Besökt 6 maj 2015]. Tillgänglig: <https://wappalyzer.com/categories/cms>

Wappalyzer. (u.å.). *Privacy Policy*. [webbsida]. [Besökt 6 maj 2015]. Tillgänglig: <https://wappalyzer.com/privacy>

Wikipedia. (2 mars 2015). *Innehållshanteringssystem*. [webbsida]. [Besökt 25 mars 2015]. Tillgänglig: <http://sv.wikipedia.org/wiki/Inneh%C3%A5llshanteringssystem>

WordPress.org:a (u.å.). *Codex - Child Themes*. [webbsida]. [Besökt 17 april 2015]. Tillgänglig: <https://codex.wordpress.org/Child_Themes>

WordPress.org:b (u.å.). *Codex - Function Reference*. [webbsida]. [Besökt 4 maj 2015]. Tillgänglig: <https://codex.wordpress.org/Function_Reference>

WordPress.org:c (u.å.). *Codex - Function Reference/add action*. [webbsida]. [Besökt 20 april 2015]. Tillgänglig: <https://codex.wordpress.org/Function_Reference/add_action>

WordPress.org:d (u.å.). *Codex - Function Reference/register sidebar*. [webbsida]. [Besökt 11 maj 2015]. Tillgänglig: <https://codex.wordpress.org/Function_Reference/register_sidebar>

WordPress.org:e (u.å.). *Codex - Function Reference/wp enqueue script*. [webbsida]. [Besökt 13 maj 2015]. Tillgänglig: <https://codex.wordpress.org/Function_Reference/wp_enqueue_script>

WordPress.org:f (u.å.). *Codex - Functions File Explained*. [webbsida]. [Besökt 4 maj 2015]. Tillgänglig: <https://codex.wordpress.org/Functions_File_Explained>

WordPress.org:g (u.å.). *Codex - Include Tags*. [webbsida]. [Besökt 3 maj 2015]. Tillgänglig: <https://codex.wordpress.org/Include_Tags>

WordPress.org:h (u.å.). *Codex - Stepping Into Template Tags*. [webbsida]. [Besökt 3 maj 2015]. Tillgänglig: <https://codex.wordpress.org/Stepping_Into_Template_Tags>

WordPress.org:i (u.å.). *Codex - Stepping into Templates*. [webbsida]. [Besökt 2 maj 2015]. Tillgänglig: <https://codex.wordpress.org/Stepping_Into_Templates>

WordPress.org:j (u.å.). *Codex - Template Hierarchy*. [webbsida]. [Besökt 9 april 2015]. Tillgänglig: <https://codex.wordpress.org/Template_Hierarchy>

WordPress.org:k (u.å.). *Codex - Theme Customization API*. [Codex]. [Besökt 20 maj 2015]. Tillgänglig: <https://codex.wordpress.org/Theme_Customization_API>

WordPress.org:l (u.å.). *Codex - WordPress Versions*. [webbsida]. [Besökt 9 april 2015]. Tillgänglig: <https://codex.wordpress.org/WordPress_Versions>

WordPress.org:m (u.å.). *Codex*. [webbsida]. [Besökt 5 juni 2015]. Tillgänglig: <https://codex.wordpress.org/Main_Page>

WordPress.org:n (u.å.). *Theme Directory - Getting Started*. [webbsida]. [Besökt 4 april 2015]. Tillgänglig: <https://wordpress.org/themes/getting-started/>

WordPress.org:o (u.å.). *Theme Directory - Popular*. [webbsida]. [Besökt 21 april 2015]. Tillgänglig: <https://wordpress.org/themes/browse/popular/>

WordPress.org:p (u.å.). *Theme Handbook - Comments. Working with Comments Template*. [webbsida]. [Besökt 19 maj 2015]. Tillgänglig: <https://developer.wordpress.org/themes/functionality/comments>

WordPress.org:q (u.å.). *Theme Handbook - Conditional Tags*. [webbsida]. [Besökt 4 maj 2015]. Tillgänglig: <https://developer.wordpress.org/themes/basics/conditional-tags>

WordPress.org:r (u.å.). *Theme Handbook - Getting Started*. [webbsida]. [Besökt 21 april 2015]. Tillgänglig: <https://developer.wordpress.org/themes/getting-started/ >

WordPress.org:s (u.å.). *Theme Handbook - Page Templates*. [webbsida]. [Besökt 30 april 2015]. Tillgänglig: <https://developer.wordpress.org/themes/basics/page-templates/>

WordPress.org:t (u.å.). *Theme Handbook - Required Template Files*. [webbsida]. [Besökt 15 maj 2015]. Tillgänglig: <https://developer.wordpress.org/themes/release/required-template-files>

WordPress.org:u (u.å.). *Theme Handbook - Sidebars*. [webbsida]. [Besökt 18 maj 2015]. Tillgänglig: <https://developer.wordpress.org/themes/functionality/sidebars>

WordPress.org:v (u.å.). *Theme Handbook - Template Files*. [webbsida]. [Besökt 30 april 2015]. Tillgänglig: <https://developer.wordpress.org/themes/basics/template-files/>

WordPress.org:w (u.å.). *Theme Handbook - Template Hierarchy*. [webbsida]. [Besökt 3 maj 2015]. Tillgänglig: <https://developer.wordpress.org/themes/basics/template-hierarchy/>

WordPress.org:x (u.å.). *Theme Handbook - Template Tags*. [webbsida]. [Besökt 4 maj 2015]. Tillgänglig: <https://developer.wordpress.org/themes/basics/template-tags>

WordPress.org:y (u.å.). *Theme Handbook - The Loop*. [webbsida]. [Besökt 6 maj 2015]. Tillgänglig: <https://developer.wordpress.org/themes/basics/the-loop/>

WordPress.org:z (u.å.). *Theme Handbook - Theme Functions*. [webbsida]. [Besökt 15 maj 2015]. Tillgänglig: <https://developer.wordpress.org/themes/basics/theme-functions/>

WordPress.org:å (u.å.). *Theme Handbook - Your First Theme*. [webbsida]. [Besökt 11 maj 2015]. Tillgänglig: <https://developer.wordpress.org/themes/getting-started/your-first-theme>

WordPress.org:ä (u.å.). *Theme Reviews - evolve*. [webbsida]. [Besökt 22 april 2015]. Tillgänglig: <https://wordpress.org/support/view/theme-reviews/evolve?filter=1>

Temats namn	Tillagd i temakatalogen	Aktuell version	har en pro-version	Angiven skapare	För	Kommentar	WooCommerce kompatibelt	Baserat på	Frameworket dyl.
ColorWay	06-aug-11	3.3.2	jo	InkThemes	Företag		nej		
Customizer	14-apr-13	3.3.19	jo	Nicholas Guillome	Företag, blogg, landing pages, forum, affärer	Udda struktur för att leta fram mallfiler, men också vad gäller funktionerna.	jo		Bootstrap
Virtue	31-jul-13	2.4.6	jo	Kandence Themes	Företag, affär, portfolio, personlig webbsdia	Många filer som hänvisar till en annnan fil. Saknas säkring för barnteman	jo		Bootstrap, Redux
Storefront	05-sep-14	1.4.3	nej	WooThemes	Webbaffär		jo	Underscores	
Zerif Lite	25-aug-14	1.7.3	jo	Themeisle	Olika typer av affärer samt portfolio	Verkar inte ha säkring för barnteman i functions.php	jo		
evolve	06-nov-10	3.2.7	jo	Theme4Press	nämns ej	PHP-koden är en stökig, blandning mellan olika syntax för villkorssatser samt oexisterande indentering. Saknas säkring för barnteman	nej		Bootstrap
Sparkling	30-	1.8.1	nej	Colorlib	Bloggar, portfolio, webbsidor		nej	Undersc	Options

Bilaga 1. Populära WordPress-teman

	mar-14							ores	Framework
Omega	26-jul-13	1.1.2	nej	Themehall	nämns ej	Är ett föräldertema	nej		
Vantage	11-sep-13	1.3.4	jo	SiteOrigin	Företag, portfolio, webbaffär		jo		
GeneratePress	19-apr-14	1.2.9.3	jo	GeneratePress	näms ej		jo	Underscores	
Enigma	25-jul-14	1.8.8	jo	Weblizar Themes	Företag, portfolio, blogg	Synes inte säkra för child-themes	jo		Bootstrap
MH Magazine	15-feb-13	1.8.0	jo	MH Themes	Tidning, nyheter, blogg		nej		
Travelify	12-jun-13	2.4.0	nej	Colorlib	Blogg, webbaffär, företag		delvis (är tillagd i functions.php, men finns ingen woocommerce.php-fil)		
Spacious	20-mar-14	1.2.5	jo	ThemeGrill	Företag, portfolio, blogg		nej		Options Framework
Moesia	15-jul-14	1.30	jo	aThemes	Företag		nej	Undderscores	Bootstrap
Responsive	12-mar-12	1.9.7.6	jo	CyberChimps	näms ej		jo		Bootstrap
ProfitMag	10-nov-14	1.0.6	nej	WP Giant	Blogg, nyhetswebbsida		nej	Underscores	
Onetone	28-feb-14	1.4.4	jo	MageeWP Themes	Företag		nej		
SKT White	30-dec-14	1.5.6	jo	SKT Themes	Företag		jo		

Bilaga 1. Populära WordPress-teman