This content is pulled from the Genesis Parent Theme’s Read Me docs.

Intro

---
title: Genesis is a powerful theme framework for WordPress
menuTitle: Home
layout: layouts/base.njk
permalink: /
tags: docs
bodyClass: home

Thousands of developers and site owners trust
[Genesis](https://www.studiopress.com/features/) as the foundation of their WordPress website.

Start using Genesis to make WordPress theme development consistent, fast, and fun today:

- Genesis features for developers
- [Genesis features for site owners](https://www.studiopress.com/features/)
- [Get Genesis now](https://www.studiopress.com/features/#genesis-feature-9)

Where to get help

This site contains technical documentation to help developers build themes and plugins for Genesis.

For setup and usage advice about StudioPress themes and plugins, refer to the [support documentation](https://my.studiopress.com/support/) or contact [StudioPress support](https://my.studiopress.com/help/) (these links require a StudioPress account). WP Engine customers can get theme documentation and support via the [WP Engine User Portal](https://my.wpengine.com/themes/docs/customization).

For help with theme alterations or code customization, join the online Genesis community groups and put your questions to the thriving Genesis community.

To make suggestions or contribute to Genesis, see the Contributing page.

Basics

---
title: Genesis Basics
menuTitle: Basics
layout: layouts/base.njk
permalink: basics/index.html
tags: docs

To make modifications to Genesis child themes, make custom themes, or write plugins that target Genesis, you'll need to learn:

- **How to use WordPress**. If you're new to WordPress, [WP101](https://www.wp101.com/) offers a great introduction to its features.
- **Basic PHP, CSS, and JavaScript**. [Know The Code](http://knowthecode.io/roadmap/genesis-developer-roadmap) offers a strong introduction to PHP, and [freeCodeCamp](https://www.freecodecamp.org/) is a great place to learn HTML, CSS and JavaScript.
- **Basic WordPress developer features**, like [hooks](https://developer.wordpress.org/plugins/hooks/), [conditional tags](https://codex.wordpress.org/Conditional_Tags), [child and parent themes](https://developer.wordpress.org/themes/advanced-topics/child-themes/) and the [template hierarchy](https://developer.wordpress.org/themes/basics/template-hierarchy/).
- **How Genesis Works**. There are some important things to know about how Genesis helps to accelerate child theme development, especially if you have built WordPress themes without Genesis before.

How Genesis Works

---
title: How Genesis Works
menuTitle: How Genesis Works

layout: layouts/base.njk
permalink: basics/how-genesis-works/index.html
tags: docs

After you understand the basics, it's helpful to know four things when building or modifying Genesis child themes:

1. Genesis outputs the HTML structure for you.
2. You control that HTML with hooks and filters.
3. You can visually inspect the position of hooks during development using a debugging plugin to help remove or add content.
4. Child themes should provide all CSS and not inherit CSS from Genesis.

1. Genesis outputs the HTML structure for you

When you create a WordPress theme without using Genesis, the [template files](https://developer.wordpress.org/themes/basics/template-files/#template-files) include HTML to define the structure of your website. To make changes to the HTML structure or attributes, you edit that HTML directly.

When you build child themes with Genesis, Genesis provides the HTML for you. Your child theme template files contain little HTML. You alter the HTML that Genesis outputs with hooks and filters. Although this is a different approach, it brings huge benefits:

Why does Genesis output HTML?

1. **You don't have to write your HTML from scratch**. Websites have similar structures, so the HTML that Genesis uses is designed to help you achieve almost any layout without major modification. Genesis brings the HTML, you bring your styling, and you still have the power to adjust or add to HTML when you need to.
2. **You benefit from years of optimization with SEO and accessibility in mind**. Genesis HTML has been refined since 2010 with the help of many contributors, as well as SEO and accessibility experts. Using Genesis saves you from making structural mistakes that could cause your site to be less accessible or harder to find in search results.
3. **You can open any Genesis child theme and see a familiar HTML structure**. This makes styling faster, especially when making changes to a Genesis child theme you have never worked with before. It also makes maintaining a range of themes more consistent, and helps to establish standards and conventions for teams of developers.

4. **You can alter HTML in one place instead of many**. Hooks and filters enable you to control the output of similar HTML wherever it appears, instead of having to update the same HTML in multiple template files.
5. **Plugins gain as much control as themes**. Plugins targeting Genesis child themes can use the same Genesis hooks and filters that Genesis child themes benefit from. This lets plugins alter HTML attributes, remove content, and inject content at specific locations in any Genesis child theme. As Genesis provides a large range of hooks throughout the HTML structure, this gives plugin developers much more flexibility to inject and alter content than they have with non-Genesis themes.

2. Control Genesis HTML with hooks and filters

Let's create a custom page template to demonstrate how to add, remove, and alter HTML in a Genesis child theme. If you would like to follow along, you'll need to [get Genesis](https://www.studiopress.com/features/#genesis-feature-9) and any Genesis child theme, such as [Genesis Sample](https://github.com/studiopress/genesis-sample/releases).

With those themes installed, activate the child theme and create a new page with a slug of 'test'.

Create a page template with Genesis

Add a page template that WordPress will use for your new 'test' page using the regular [WordPress page template hierarchy](https://developer.wordpress.org/themes/template-files-section/page-template-files/#page-templates-within-the-template-hierarchy), like this:

1. Create a file named `page-test.php` in the root of your child theme.
2. Leave its contents blank for now.

Visit the URL of the 'test' page you created. You should see a blank page.

If you were not using Genesis, you'd now have to construct your HTML in that file, perhaps by copying it from other files or by outputting content and modifications around a skeleton of header and footer includes.

With Genesis, you only need to add this code to your `page-test.php` file:

```php
<?php
genesis();
```

Refresh the 'test' page URL again. You'll now see the full site and page content.

What just happened?

When you call `genesis()`, the Genesis Framework determines your site HTML from:

- The default HTML structure in the Genesis parent theme.
- Hooks and filters you've added in your child theme and plugins that change the HTML output.
- User preferences in Genesis settings that alter how the site is displayed.

Let's now see how to influence the HTML for our custom page template.

Add HTML content via a hook

To add some HTML displaying a notice immediately below the page title, modify your `page-test.php` template to 'hook' a function to the `genesis_before_entry_content` action:

```php
<?php

add_action( 'genesis_before_entry_content', 'theme_prefix_show_notice' );
/**
 * Display a custom notice.
 */
function theme_prefix_show_notice() {
	echo '<p class="notice">This page has a custom template.</p>';
}

genesis();
```

You'll see the HTML you wanted to output above your regular page content.

Genesis provides a wide range of other actions you can hook content to. Discover them from the Genesis hooks documentation or by using a plugin to visualize hook names and positions.

You can use the tips we're applying to this custom page template to add, remove, and adjust content throughout your site. You can also hook and unhook actions in your <code>functions.php</code> file, where you don't need to include the <code>genesis()</code> call. You can use WordPress conditional tags to determine which pages to affect.

Remove HTML content by unhooking it

You can also remove content that Genesis added. For example, you can unhook the functions that Genesis used to add the title content with this in your template file:

```php
<?php
remove_action( 'genesis_entry_header', 'genesis_entry_header_markup_open', 5 );
remove_action( 'genesis_entry_header', 'genesis_do_post_title', 10 );
remove_action( 'genesis_entry_header', 'genesis_entry_header_markup_close', 12 );

genesis();
```

To discover which actions and functions to unhook to remove other content, we recommend browsing Genesis source code or using a plugin to find hook names and positions.

Alter page settings and HTML attributes using filters

You can use [WordPress filters](https://developer.wordpress.org/plugins/hooks/filters/) to alter page settings or modify existing code such as HTML attributes.

For example, to force a full-width page layout, you can add this filter above the `genesis()` call:

```php
add_filter( 'genesis_site_layout', '__genesis_return_full_width_content' );
```

Or to add an ID attribute to the `site-container` div element, you can do this:

```php
add_filter( 'genesis_attr_site-container', 'theme_prefix_container_attributes' );
/**
 * Modify the site-container div attributes.
 *
 * @param array $attr The original attributes.
 * @return array The modified attributes.
 */
function theme_prefix_container_attributes( $attr ) {
	$attr['id'] = 'site-container';
```

```
	return $attr;
}
```

Find other Genesis filters here and in the [StudioPress snippets](https://my.studiopress.com/documentation/snippets/).

You can still build templates from scratch without using the HTML Genesis provides if your business needs demand it. Omit the <code>genesis()</code> function from your template file and provide your own HTML instead. Note that Genesis filters and actions that usually affect HTML structure or attributes will no longer function for that template. Bypassing Genesis HTML output is best used sparingly for special cases, such as custom landing page templates that may require no Genesis features.

3. Inspect the position of hooks

How do you know what hooks existing content is attached to, and what hooks to use to add new content?

You could read the Genesis source code, but it's often faster to use a plugin to help visualize hooks available on the page you want to modify. There are several third-party plugins to help with this:

- [Simply Show Hooks](https://wordpress.org/plugins/simply-show-hooks/)
- [Genesis Visual Hook Guide](https://wordpress.org/plugins/genesis-visual-hook-guide/)
- [Hooks Visualizer](https://wordpress.org/plugins/hooks-visualizer/)

4. Child themes should provide their own CSS

You should use your own styles in your child theme's `style.css` file, and not inherit the parent Genesis styles.

The Genesis parent theme `style.css` file is not intended to be inherited by the child theme, and may be removed or altered in future versions of Genesis.

Genesis automatically enqueues the `style.css` file from your Genesis child theme folder.

Where to go from here

- Learn about Genesis Developer features like theme supports and the Onboarding API.
- [Get Genesis](https://www.studiopress.com/features/#genesis-feature-9), then download and modify an existing Genesis child theme, such as [Genesis Sample](https://github.com/studiopress/genesis-sample/releases).
- Join the Genesis community to ask questions and get help.
- Contribute to Genesis and help to shape its future.

Genesis Hooks

---
title: Genesis Hooks
menuTitle: Genesis Hooks
layout: layouts/base.njk
permalink: basics/genesis-hooks/index.html
tags: docs

Genesis Hooks are currently documented in your StudioPress account area:

View Genesis Hooks

WP Engine customers who do not have a StudioPress account can find hook documentation in the [WP Engine User Portal](https://my.wpengine.com/themes/docs/customization/guides-and-references/hook-reference/).

We intend to migrate hook information here and ensure all hooks are documented.

Genesis Filters

---
title: Genesis Filters
menuTitle: Genesis Filters
layout: layouts/base.njk

permalink: basics/genesis-filters/index.html
tags: docs

Genesis Filters are currently documented in your StudioPress account area:

View Genesis Filters

WP Engine customers who do not have a StudioPress account can find filter documentation in the [WP Engine User Portal](https://my.wpengine.com/themes/docs/filters/genesis-filters/filter-reference/).

We intend to migrate filter information here and ensure all filters are documented.

Privacy

---
title: Genesis Privacy
menuTitle: Privacy
layout: layouts/base.njk
permalink: basics/privacy/index.html
minVersion: Genesis 2.7.0+ and WordPress 4.9.6+
tags: docs

Developers and site owners can use Genesis without negatively affecting user privacy. Genesis makes use of WordPress personal data export and deletion features to help site owners remove or provide personal data in response to a privacy request.

Genesis user data
Genesis may store personally identifying information about users if they provide personal data in their user profiles. This data is stored in the WordPress database with other user profile information that users provide. It is not sent or stored additionally elsewhere.

Genesis refers to this user profile data as:
- Author Archive Settings
- Author Archive SEO Settings

These settings can be found on the user profile edit screen in the WordPress admin area.

If a user does not provide personally identifying data, Genesis does not store anything that is personally identifying about that user.

Genesis also offers the option to email someone when a new version of Genesis is available. This option allows an email address to be stored, which appears in the theme settings page under Genesis → Theme Settings and in the Customizer at Theme Settings → Updates. This email will be deleted if it matches the address of a user who has requested their data to be removed.

WordPress privacy tools
WordPress offers privacy tools for site owners, accessible from the Tools menu in the WordPress admin area:
- **Export Personal Data**: allows admin users to export a user's personal data.
- **Erase Personal Data**: allows admin users to erase a user's personal data.

When someone uses these tools, Genesis informs WordPress of the data mentioned above so that WordPress can include it when processing the privacy request.

For example, here is the output of an Erase Personal Data action, showing that user-provided data relating to Genesis features was successfully erased upon request:

Features

---
title: Genesis Developer Features
menuTitle: Developer Features
layout: layouts/base.njk
permalink: developer-features/index.html
tags: docs

Genesis enhances WordPress with powerful features that site owners love.

For your clients

Developers using Genesis can offer their clients these features without having to build them from scratch or install additional plugins:

- **Selectable page and archive layouts**, including full-width, content-sidebar, sidebar-content and more.
- **Accessible HTML markup** and other accessibility considerations.
- **Great SEO by default**, with schema.org markup and SEO options that require no additional plugins unless you choose to use your own.
- **Custom widgets** for User Profile, Features Posts, and Featured Page.
- **Custom meta boxes** such as the Scripts field, enabling per-page JavaScript for conversion analytics and more.
- **A range of extendable theme options** such as how archives are displayed.
- **Gutenberg Optimized child themes** that you can build upon and adapt. See [Genesis child themes](https://my.studiopress.com/themes/).

The Tenon accessibility team found Genesis-powered WordPress sites are the most accessible in their survey of content management systems and frameworks.

Discover Genesis on studiopress.com

For developers

Genesis offers features for busy developers too:

- **Genesis provides a battle-tested HTML structure by default** to help you avoid SEO and accessibility pitfalls when hand-writing HTML from scratch, and to accelerate the development cycle.
- **The Genesis Onboarding API is a theme setup wizard** for Genesis child themes that enables you to install plugins and set up sample homepage and page content upon theme activation.
- **The Genesis Configuration API** lets you override Genesis configuration in your child theme, and store your own PHP configuration centrally in your theme's `config` folder.
- **Genesis offers a wealth of theme support options** to enhance your custom child theme with Genesis features. These also simplify the addition of common site functionality such as footer and after-post widget areas.

- **A starter child theme called [Genesis Sample](https://github.com/studiopress/genesis-sample/releases)** to use as a starting point for your Genesis powered sites.
- **Constants**, filters, actions, and helper functions to control the output of Genesis child themes with minimal and maintainable code.
- **A vibrant Genesis community** to get help and share your feedback.
- **A documented contribution process** via GitHub.
- **A release history dating back to January 2010** and a bright roadmap.

Learn what you need to develop with Genesis, put your questions to the Genesis community, or get Genesis now:

Get Genesis

Theme Support

---
title: Genesis Theme Support
menuTitle: Theme Support
layout: layouts/base.njk
permalink: developer-features/theme-support/index.html
tags: docs

Default theme supports

Genesis adds the following [theme supports](https://developer.wordpress.org/reference/functions/add_theme_support/) to Genesis child themes by default:

```php
add_theme_support( 'menus' );
add_theme_support( 'post-thumbnails' );
add_theme_support( 'title-tag' );
add_theme_support( 'automatic-feed-links' );
add_theme_support( 'genesis-inpost-layouts' );
```

```
add_theme_support( 'genesis-archive-layouts' );
add_theme_support( 'genesis-admin-menu' );
add_theme_support( 'genesis-seo-settings-menu' );
add_theme_support( 'genesis-import-export-menu' );
add_theme_support( 'genesis-customizer-theme-settings' );
add_theme_support( 'genesis-customizer-seo-settings' );
add_theme_support( 'genesis-auto-updates' );
add_theme_support( 'genesis-breadcrumbs' );
```

These enable the following features:

<table>
  <tr>
    <th>Feature</th>
    <th>Description</th>
  </tr>
  <tr>
    <td>menus</td>
    <td>WordPress menus.</td>
  </tr>
  <tr>
    <td>post-thumbnails</td>
    <td>See post thumbnails.</td>
  </tr>
  <tr>
    <td>title-tag</td>
    <td>See title tag.</td>
  </tr>
  <tr>
    <td>automatic-feed-links</td>
    <td>See automatic feed
links.</td>
  </tr>
  <tr>
    <td>genesis-inpost-layouts</td>
    <td>Genesis layouts for posts and pages.</td>
  </tr>
  <tr>
    <td>genesis-archive-layouts</td>
    <td>Genesis layouts on archives.</td>
  </tr>
  <tr>

```
    <td>genesis-admin-menu</td>
    <td>Displays the Genesis menu.</td>
  </tr>
  <tr>
    <td>genesis-seo-settings-menu</td>
    <td>Displays the Genesis SEO settings menu.</td>
  </tr>
  <tr>
    <td>genesis-import-export-menu</td>
    <td>Displays the Genesis import/export menu.</td>
  </tr>
  <tr>
    <td>genesis-customizer-theme-settings</td>
    <td>Adds Genesis theme settings to the Customizer.</td>
  </tr>
  <tr>
    <td>genesis-customizer-seo-settings</td>
    <td>Adds Genesis SEO settings to the Customizer.</td>
  </tr>
  <tr>
    <td>genesis-auto-updates</td>
    <td>Adds a UI option to enable Genesis update checks.</td>
  </tr>
  <tr>
    <td>genesis-breadcrumbs</td>
    <td>Genesis breadcrumb features and options.</td>
  </tr>
</table>

To opt-out of these, you can remove support in your child theme's `functions.php` with the [`remove_theme_support()`](https://developer.wordpress.org/reference/functions/remove_theme_support/) function:

```php
remove_theme_support( 'automatic-feed-links' );
```

Opt-in theme supports

You can add support for additional WordPress and Genesis features using the [`add_theme_support()`](https://developer.wordpress.org/reference/functions/add_theme_support/) WordPress function in your child theme's `functions.php` file.
```

HTML5

We strongly recommend that you add the WordPress HTML5 theme support to ensure Genesis outputs HTML5 and not XHTML:

```php
add_theme_support(
	'html5',
	array(
		'caption',
		'comment-form',
		'comment-list',
		'gallery',
		'search-form',
	)
);
```

Genesis accessibility

Add support for Genesis accessibility features (also strongly recommended).

```php
add_theme_support(
	'genesis-accessibility',
	array(
		'404-page',
		'drop-down-menu',
		'headings',
		'search-form',
		'skip-links',
	)
);
```

<table>
  <tr>
    <th>Option</th>
    <th>Description</th>
  </tr>
  <tr>
    <td>404-page</td>
    <td>Add an additional heading and adjust heading levels on the 404 page.</td>

```
  </tr>
  <tr>
    <td>drop-down-menu</td>
    <td>Add scripts to improve accessibility of drop-down menus.</td>
  </tr>
  <tr>
    <td>headings</td>
    <td>Add additional headings for screen reader users to aid navigation.</td>
  </tr>
  <tr>
    <td>search-form</td>
    <td>Improve search form labels.</td>
  </tr>
  <tr>
    <td>skip-links</td>
    <td>Add <a href="https://webaim.org/techniques/skipnav/">skip links</a> markup.</td>
  </tr>
</table>
```

Genesis menus

Add a Primary and Secondary navigation menu with given names:

```php
add_theme_support(
	'genesis-menus',
	array(
		'primary'   => __( 'Primary Menu', 'genesis-sample' ),
		'secondary' => __( 'Secondary Menu', 'genesis-sample' ),
	)
);
```

Genesis structural wraps

Add `div` elements with a `.wrap` class to wrap HTML that Genesis outputs. This can assist with styling for full-width layouts.

```php
add_theme_support(
	'genesis-structural-wraps',
	array(
		'header',
```

```
                'menu-primary',
                'menu-secondary',
                'footer-widgets',
                'footer'
        )
);
```

Genesis responsive viewport

Add a viewport meta tag for responsive display on mobile browsers.

```php
add_theme_support( 'genesis-responsive-viewport' );
```

Genesis after entry widget area

Add a widget area after post entries. Useful for calls to action.

```php
add_theme_support( 'genesis-after-entry-widget-area' );
```

Genesis footer widget areas

Add the given number of footer widget areas. You must provide your own CSS to control the layout of these.

```php
add_theme_support( 'genesis-footer-widgets', 3 );
```

Support for Genesis features in custom post types

Genesis automatically adds these features to the Post and Page type:

<table>
  <tr>
    <th>Option</th>
    <th>Description</th>
  </tr>
  <tr>

```
    <td>genesis-seo</td>
    <td>Add a Genesis SEO panel to your post type.</td>
  </tr>
  <tr>
    <td>genesis-scripts</td>
    <td>Add a Genesis Scripts field to your post type for per-page scripts.</td>
  </tr>
  <tr>
    <td>genesis-layouts</td>
    <td>Add Genesis layout options to your post type.</td>
  </tr>
</table>
```

You can extend these features to your own custom post types using [`add_post_type_support()`](https://developer.wordpress.org/reference/functions/add_post_type_support/):

```php
add_post_type_support( 'your-custom-post-type-slug', array( 'genesis-seo', 'genesis-scripts', 'genesis-layouts' ) );
```

Onboarding

```
---
title: Genesis Theme Support
menuTitle: Theme Support
layout: layouts/base.njk
permalink: developer-features/theme-support/index.html
tags: docs
---
```

Default theme supports

Genesis adds the following [theme supports](https://developer.wordpress.org/reference/functions/add_theme_support/) to Genesis child themes by default:

```php
add_theme_support( 'menus' );
add_theme_support( 'post-thumbnails' );
```

```
add_theme_support( 'title-tag' );
add_theme_support( 'automatic-feed-links' );
add_theme_support( 'genesis-inpost-layouts' );
add_theme_support( 'genesis-archive-layouts' );
add_theme_support( 'genesis-admin-menu' );
add_theme_support( 'genesis-seo-settings-menu' );
add_theme_support( 'genesis-import-export-menu' );
add_theme_support( 'genesis-customizer-theme-settings' );
add_theme_support( 'genesis-customizer-seo-settings' );
add_theme_support( 'genesis-auto-updates' );
add_theme_support( 'genesis-breadcrumbs' );
```

These enable the following features:

<table>
  <tr>
    <th>Feature</th>
    <th>Description</th>
  </tr>
  <tr>
    <td>menus</td>
    <td>WordPress menus.</td>
  </tr>
  <tr>
    <td>post-thumbnails</td>
    <td>See post thumbnails.</td>
  </tr>
  <tr>
    <td>title-tag</td>
    <td>See title tag.</td>
  </tr>
  <tr>
    <td>automatic-feed-links</td>
    <td>See automatic feed
links.</td>
  </tr>
  <tr>
    <td>genesis-inpost-layouts</td>
    <td>Genesis layouts for posts and pages.</td>
  </tr>
  <tr>
    <td>genesis-archive-layouts</td>

```
    <td>Genesis layouts on archives.</td>
  </tr>
  <tr>
    <td>genesis-admin-menu</td>
    <td>Displays the Genesis menu.</td>
  </tr>
  <tr>
    <td>genesis-seo-settings-menu</td>
    <td>Displays the Genesis SEO settings menu.</td>
  </tr>
  <tr>
    <td>genesis-import-export-menu</td>
    <td>Displays the Genesis import/export menu.</td>
  </tr>
  <tr>
    <td>genesis-customizer-theme-settings</td>
    <td>Adds Genesis theme settings to the Customizer.</td>
  </tr>
  <tr>
    <td>genesis-customizer-seo-settings</td>
    <td>Adds Genesis SEO settings to the Customizer.</td>
  </tr>
  <tr>
    <td>genesis-auto-updates</td>
    <td>Adds a UI option to enable Genesis update checks.</td>
  </tr>
  <tr>
    <td>genesis-breadcrumbs</td>
    <td>Genesis breadcrumb features and options.</td>
  </tr>
</table>

To opt-out of these, you can remove support in your child theme's `functions.php` with the [`remove_theme_support()`](https://developer.wordpress.org/reference/functions/remove_theme_support/) function:

```php
remove_theme_support( 'automatic-feed-links' );
```

Opt-in theme supports
```

You can add support for additional WordPress and Genesis features using the [`add_theme_support()`](https://developer.wordpress.org/reference/functions/add_theme_support/) WordPress function in your child theme's `functions.php` file.

HTML5

We strongly recommend that you add the WordPress HTML5 theme support to ensure Genesis outputs HTML5 and not XHTML:

```php
add_theme_support(
	'html5',
	array(
		'caption',
		'comment-form',
		'comment-list',
		'gallery',
		'search-form',
	)
);
```

Genesis accessibility

Add support for Genesis accessibility features (also strongly recommended).

```php
add_theme_support(
	'genesis-accessibility',
	array(
		'404-page',
		'drop-down-menu',
		'headings',
		'search-form',
		'skip-links',
	)
);
```

<table>
  <tr>
    <th>Option</th>
    <th>Description</th>

```
  </tr>
  <tr>
    <td>404-page</td>
    <td>Add an additional heading and adjust heading levels on the 404 page.</td>
  </tr>
  <tr>
    <td>drop-down-menu</td>
    <td>Add scripts to improve accessibility of drop-down menus.</td>
  </tr>
  <tr>
    <td>headings</td>
    <td>Add additional headings for screen reader users to aid navigation.</td>
  </tr>
  <tr>
    <td>search-form</td>
    <td>Improve search form labels.</td>
  </tr>
  <tr>
    <td>skip-links</td>
    <td>Add <a href="https://webaim.org/techniques/skipnav/">skip links</a> markup.</td>
  </tr>
</table>

### Genesis menus

Add a Primary and Secondary navigation menu with given names:

```php
add_theme_support(
	'genesis-menus',
	array(
		'primary'   => __( 'Primary Menu', 'genesis-sample' ),
		'secondary' => __( 'Secondary Menu', 'genesis-sample' ),
	)
);
```

Genesis structural wraps

Add `div` elements with a `.wrap` class to wrap HTML that Genesis outputs. This can assist with styling for full-width layouts.

```php
```
```

```
add_theme_support(
	'genesis-structural-wraps',
	array(
		'header',
		'menu-primary',
		'menu-secondary',
		'footer-widgets',
		'footer'
	)
);
```

Genesis responsive viewport

Add a viewport meta tag for responsive display on mobile browsers.

```php
add_theme_support( 'genesis-responsive-viewport' );
```

Genesis after entry widget area

Add a widget area after post entries. Useful for calls to action.

```php
add_theme_support( 'genesis-after-entry-widget-area' );
```

Genesis footer widget areas

Add the given number of footer widget areas. You must provide your own CSS to control the layout of these.

```php
add_theme_support( 'genesis-footer-widgets', 3 );
```

Support for Genesis features in custom post types

Genesis automatically adds these features to the Post and Page type:

<table>
  <tr>

```
    <th>Option</th>
    <th>Description</th>
  </tr>
  <tr>
    <td>genesis-seo</td>
    <td>Add a Genesis SEO panel to your post type.</td>
  </tr>
  <tr>
    <td>genesis-scripts</td>
    <td>Add a Genesis Scripts field to your post type for per-page scripts.</td>
  </tr>
  <tr>
    <td>genesis-layouts</td>
    <td>Add Genesis layout options to your post type.</td>
  </tr>
</table>
```

You can extend these features to your own custom post types using [`add_post_type_support()`](https://developer.wordpress.org/reference/functions/add_post_type_support/):

```php
add_post_type_support( 'your-custom-post-type-slug', array( 'genesis-seo', 'genesis-scripts', 'genesis-layouts' ) );
```

Configuration

```
---
title: Genesis Configuration
menuTitle: Configuration
layout: layouts/base.njk
permalink: developer-features/configuration/index.html
tags: docs
minVersion: Genesis 2.8.0+
---
```

The Genesis configuration API lets Genesis child theme developers do two things:

1. [Override certain Genesis parent theme settings](#override-genesis-features).
2. [Load configuration data](#load-child-theme-settings-from-your-theme's-config-folder) from the child theme's `config` folder.

To use these features, create the <code>config</code> folder in the root of your child theme (at the same level as <code>style.css</code>) if it does not exist already.

Override Genesis features

The following config files from the Genesis parent theme can be overridden by placing a file of the same name in your child theme's `config` folder.

- `genesis/config/breadcrumbs.php`
- `genesis/config/customizer-seo-settings.php`
- `genesis/config/customizer-theme-settings.php`
- `genesis/config/layouts.php`

You can copy the code in these files to your child theme, then make desired changes.

For example, to alter the layouts your child theme offers so that it only includes a “full-width” and “content-sidebar” layout instead of the six layouts Genesis offers by default, you can create a file at `your-child-theme/config/layouts.php` with the following content:

```php
<?php
/**
 * Your Theme Name
 *
 * Overrides `genesis/config/layouts.php` to set default theme layouts.
 *
 * @package Theme Name
 * @author  Your Name
 * @license GPL-2.0-or-later
 * @link    https://example.com/
 */

// Path to layout images in the Genesis parent theme.
$url = GENESIS_ADMIN_IMAGES_URL . '/layouts/';

return array(
	'content-sidebar'         => array(
		'label'   => __( 'Content, Primary Sidebar', 'your-theme-slug' ),
		'img'     => $url . 'cs.gif',
		'default' => is_rtl() ? false : true,
```

```
                'type'    => array( 'site' ),
        ),
        'full-width-content'      => array(
                'label' => __( 'Full Width Content', 'your-theme-slug' ),
                'img'   => $url . 'c.gif',
                'type'  => array( 'site' ),
        ),
);
```

Another approach is to use the child theme config files to load parent theme config, then add and unset default values instead of reproducing the parent config file contents explicitly. For example, the same result above can be achieved with a `your-child-theme/config/layouts.php` file that looks like this:

```php
<?php
/**
 * Your Theme Name
 *
 * Overrides `genesis/config/layouts.php` to set default theme layouts.
 *
 * @package Theme Name
 * @author  Your Name
 * @license GPL-2.0-or-later
 * @link    https://example.com/
 */

$layouts = array();

$genesis_layouts_config = get_template_directory() . '/config/layouts.php';

if ( is_readable( $genesis_layouts_config ) ) {
        $layouts = require $genesis_layouts_config;
        unset( $layouts['sidebar-content'] );
        unset( $layouts['content-sidebar-sidebar'] );
        unset( $layouts['sidebar-sidebar-content'] );
        unset( $layouts['sidebar-content-sidebar'] );
}

return $layouts;
```

Load child theme settings from your theme's config folder

Genesis 2.8.0+ includes a `genesis_get_config()` function. This allows you to fetch custom configuration data from your child theme's `config` folder.

Child themes contain PHP configuration data — the data that makes your theme different to other themes — that is scattered across PHP files. Your `functions.php` file might include code that looks like this, for example:

```php
add_theme_support(
	'custom-logo',
	array(
		'height'      => 120,
		'width'       => 700,
		'flex-height' => true,
		'flex-width'  => true,
	)
);
```

And you may have a separate file that sets up WordPress block editor features, such as custom font sizes:

```php
add_theme_support(
	'editor-font-sizes',
	array(
		array(
			'name'      => __( 'Small', 'your-theme-slug' ),
			'shortName' => __( 'S', 'your-theme-slug' ),
			'size'      => 12,
			'slug'      => 'small',
		),
		array(
			'name'      => __( 'Normal', 'your-theme-slug' ),
			'shortName' => __( 'M', 'your-theme-slug' ),
			'size'      => 16,
			'slug'      => 'normal',
		),
		array(
			'name'      => __( 'Large', 'your-theme-slug' ),
			'shortName' => __( 'L', 'your-theme-slug' ),
```

```
                'size'      => 20,
                'slug'      => 'large',
            ),
            array(
                'name'      => __( 'Larger', 'your-theme-slug' ),
                'shortName' => __( 'XL', 'your-theme-slug' ),
                'size'      => 24,
                'slug'      => 'larger',
            ),
        )
);
```

With `genesis_get_config()`, you can instead write code like this:

```php
add_theme_support( 'custom-logo', genesis_get_config( 'custom-logo' ) );
```

With a file at `your-theme-name/config/custom-logo.php` that looks like this:

```php
return array(
        'height'      => 120,
        'width'       => 700,
        'flex-height' => true,
        'flex-width'  => true,
);
```

The `editor-font-sizes` theme support becomes:

```php
add_theme_support( 'editor-font-sizes', genesis_get_config( 'editor-font-sizes' ) );
```

With a file at `your-theme-name/config/editor-font-sizes.php` that looks like this:

```php
return array(
        array(
            'name'      => __( 'Small', 'your-theme-slug' ),
            'shortName' => __( 'S', 'your-theme-slug' ),
```

```
                'size'      => 12,
                'slug'      => 'small',
        ),
        array(
                'name'      => __( 'Normal', 'your-theme-slug' ),
                'shortName' => __( 'M', 'your-theme-slug' ),
                'size'      => 16,
                'slug'      => 'normal',
        ),
        array(
                'name'      => __( 'Large', 'your-theme-slug' ),
                'shortName' => __( 'L', 'your-theme-slug' ),
                'size'      => 20,
                'slug'      => 'large',
        ),
        array(
                'name'      => __( 'Larger', 'your-theme-slug' ),
                'shortName' => __( 'XL', 'your-theme-slug' ),
                'size'      => 24,
                'slug'      => 'larger',
        ),
);
```

Moving your configuration data to the `config` folder in this way is optional. You can use it for all PHP data in your theme, for select data, or not at all. The advantages of this approach are:

- **It puts theme configuration in a single place.** Instead of being spread throughout the theme in disparate PHP files, the bulk of what makes your theme unique outside of its CSS will be stored in a single folder. The benefits of this include better readability and maintainability.

- **It opens the door to new tooling.** This includes theme generators (where a website generates a custom build of a theme by writing new config files that match user preferences), as well as marketing pages that can automatically generate a list of theme features by skimming content from each theme's config folder. Although these tools are already possible with configuration spread throughout a theme, centralizing config makes this easier.

<p class="notice-small">
The parent theme config file names are reserved for use by Genesis. You should not name a config file <code>breadcrumbs.php</code>, for example, unless you intend to override Genesis breadcrumbs configuration. The <code>onboarding.php</code> filename is also reserved for use in the Onboarding API.

</p>

Constants

---
title: Genesis Constants
menuTitle: Constants
layout: layouts/base.njk
permalink: developer-features/constants/index.html
tags: docs

Directory and URL constants

These constants are available to child theme developers for convenience and performance:

<table>
  <tr>
    <th>Constant</th>
    <th>Equivalent</th>
  </tr>
  <tr>
    <td><code>PARENT_DIR</code></td>
    <td><a
href="https://codex.wordpress.org/Function_Reference/get_template_directory"><code>get_tem
plate_directory()</code></td>
  </tr>
  <tr>
    <td><code>CHILD_DIR</code></td>
    <td><a
href="https://codex.wordpress.org/Function_Reference/get_stylesheet_directory"><code>get_st
ylesheet_directory()</code></td>
  </tr>
  <tr>
    <td><code>PARENT_URL</code></td>
    <td><a
href="https://codex.wordpress.org/Function_Reference/get_template_directory_uri"><code>get_
template_directory_uri()</code></td>
  </tr>
  <tr>
    <td><code>CHILD_URL</code></td>

```
    <td><a
href="https://codex.wordpress.org/Function_Reference/get_stylesheet_directory_uri"><code>ge
t_stylesheet_directory_uri()</code></a></td>
  </tr>
</table>
```

You can use the code in the *Constant* column wherever you might use the code in the *Equivalent* column in your child theme. So instead of:

```php
wp_enqueue_script(
	'custom-theme',
	get_stylesheet_directory_uri() . '/js/custom-theme.js',
	array( 'jquery' ),
	wp_get_theme()->get( 'Version' ),
	true
);
```

You can do this for brevity and to avoid an additional function call:

```php
wp_enqueue_script(
	'custom-theme',
	CHILD_URL . '/js/custom-theme.js', // <-- Constant used here.
	array( 'jquery' ),
	wp_get_theme()->get( 'Version' ),
	true
);
```

Additional constants

Genesis uses additional constants that are mostly useful for those contributing to Genesis itself:

```
<table>
  <tr>
    <th>Constant</th>
    <th>Example Values</th>
  </tr>
  <tr>
    <td><code>PARENT_THEME_NAME</code></td>
    <td>Genesis</td>
```

```
  </tr>
  <tr>
    <td><code>PARENT_THEME_VERSION</code></td>
    <td>2.8.0, 2.8.0-beta2</td>
  </tr>
  <tr>
    <td><code>PARENT_THEME_BRANCH</code></td>
    <td>2.8</td>
  </tr>
  <tr>
    <td><code>GENESIS_IMAGES_URL</code></td>
    <td>https://example.com/wp-content/themes/genesis/images</td>
  </tr>
  <tr>
    <td><code>GENESIS_ADMIN_IMAGES_URL</code></td>
    <td>https://example.com/wp-content/themes/genesis/lib/admin/images</td>
  </tr>
  <tr>
    <td><code>GENESIS_CSS_URL</code></td>
    <td>https://example.com/wp-content/themes/genesis/lib/css</td>
  </tr>
  <tr>
    <td><code>GENESIS_VIEWS_DIR</code></td>
    <td>/path/to/site/wp-content/themes/genesis/lib/views</td>
  </tr>
  <tr>
    <td><code>GENESIS_CONFIG_DIR</code></td>
    <td>/path/to/site/wp-content/themes/genesis/config</td>
  </tr>
  <tr>
    <td><code>GENESIS_SETTINGS_FIELD</code></td>
    <td>genesis-settings</td>
  </tr>
  <tr>
    <td><code>GENESIS_SEO_SETTINGS_FIELD</code></td>
    <td>genesis-seo-settings</td>
  </tr>
  <tr>
    <td><code>GENESIS_CPT_ARCHIVE_SETTINGS_FIELD_PREFIX</code></td>
    <td>genesis-cpt-archive-settings-</td>
  </tr>
</table>
```

Testing for Genesis features

It is generally better to test for a specific function or class rather than using the Genesis version constants. We recommend this:

```php
if ( function_exists( 'genesis_get_config' ) ) {
	// `genesis_get_config()` exists and is safe to use.
}
```

Over comparisons like this:

```php
if ( version_compare( PARENT_THEME_VERSION, '2.8.0', '>=' ) ) {
    // Genesis version is 2.8.0 or higher.
}
```

Checking for the function by name ensures your code will not throw a fatal error if that function is deprecated and removed in a future version of Genesis.

Genesis Community

---
title: The Genesis Community
menuTitle: Community
layout: layouts/base.njk
permalink: contribute/community/index.html
tags: docs

Learn from and connect with other developers, designers, and site owners who use and love the Genesis Framework.

GenesisWP Slack workspace

The Genesis Slack community is one of the best places to get live insights from a talented group of Genesis developers, designers, contributors and fans. You'll also find Genesis lead developers and the StudioPress team hanging out here.

Join GenesisWP Slack

StudioPress Community Forum

Connect with others in the StudioPress community to discuss WordPress, StudioPress theme customization, web design and more. Just choose a new username and log in.

Visit the StudioPress forum

Genesis Facebook group

Join a large group of StudioPress users on Facebook to ask questions, share tips and keep track of community news and events.

Join the Facebook group

Help from recommended developers

Need paid help with a customization issue? Check our recommended developers list. (Joining this list is by invitation only.)

See recommended developers