Introduction

- Tutorial theme
- A view
- Make it dynamic
- The context

Let’s get familiar with some of the concepts of Timber and Twig.

Tutorial theme

If you want to start from scratch, you can use the following command. Run it from the **wp-content/themes** folder of your WordPress installation.

```
composer create-project timber/learn-timber-theme learn-timber-theme
```

And now …

1. Go to your WordPress Admin and activate the new theme under **Design** → **Themes**.
2. Create a page with any title you like.
3. Select the newly created page as the **Homepage** under **Settings** → **Reading**.

A view

You’re maybe used to set up the template file for a single post like this:

single.php

```
<h1><?php the_title(); ?></h1>
```

This template displays a post’s title by calling the PHP function `the_title()`. In Timber, we don’t do it like that. Instead, we make use of the Twig templating engine.

The goal of a templating engine is to separate your logic from your HTML templates. When working with Timber, you'll want to prepare the data you'll need for your templates in PHP first – and then pass it to a Twig template. We call the Twig templates "views".

To render a view, you can use `Timber::render()`.

index.php

```
Timber::render('index.twig');
```

This will look for an **index.twig** file in the **views** folder of your theme and render the contents of that template.

index.twig

```
<h1>A Timber Tutorial</h1>
```

We don't have any data yet. Let's create an array with data that we then pass to our view with the second parameter for `Timber::render()`.

```
$data = [
    'title' => 'A Timber Tutorial',
];

Timber::render('index.twig', $data);
```

index.twig

```
<h1>{{ title }}</h1>
```

Our associative `$data` array was turned into variables that we can use directly in our Twig template.

The `{{ }}` delimiters are more or less the same as the `echo` command in PHP. Whenever you want to output a variable or an expression in Twig, use the double curly braces. You can also see that when we reference variables in Twig, we don't need a `$` like we do in PHP.

Make it dynamic

The `title` variable is still a static string. Let's make it a little more dynamic by using the `get_the_title()` function instead of a static title to get the title of the current WordPress post.

index.php

```
$data = [
    'title' => get_the_title(),
];

Timber::render('index.twig', $data);
```

See how there's no HTML in our PHP file? And do you see how we fetch the data we need in PHP and output it in Twig? This is an important concept called *separation of concerns*.

The context

In Timber, we prepare a lot for you in the background, so you don't have to remember function names like `get_the_title()`.

Most Timber templates look like this:

index.php

```
$context = Timber::context();

Timber::render('index.twig', $context);
```

What happens here? In Timber, you can use the `Timber::context()` function to get **an array of data that you need in most of your templates**. It includes things like the site name, the site description or the navigation menu that you probably need in every template. You can read more about this in the Context Guide whenever you're ready.

Timber always prepares some of the data you will need behind the scenes. And for a singular post template, this will be a `post` variable that holds information about your post.

index.twig

```
<h1>{{ post.title }}</h1>
```

The `title` is a property (or a method, but more on that later) of `post`. And you can access properties in Twig by using dots.

Let's keep improving. Continue reading about Template Inheritance and Includes in the next section.

□□□ #2
□ □□□ □□□ 2026-08-30 00:14:01 UTC
□ □□□ □□□ 2026-08-30 00:14:01 UTC