Date/Time in Timber

- Working with dates and times in WordPress
  - WordPress and timezones
- Creating dates from strings
  - Create a date from a timestamp
  - Create a date from a date string
    - Dates without times
    - strtotime()
- Control the date display format
  - Post dates
- Twig filters and functions
- Current date
- Time differences
- Comparing dates

Working with dates and times in WordPress

Before we tell you how to work with dates and times in Timber, we need to look at how WordPress handles date and time.

When you work with dates and times in a WordPress context, it's best if you stick to the date and time functionality that WordPress provides for you. Timber tries to use the default functionality as much as it can. To prepare your environment, make sure to check the following WordPress settings:

- Set the correct timezone in *Settings* → *General*.
- Set the desired date and time formats in *Settings* → *General*. You can change the format whenever you display a date later.

In WordPress 5.3, there were improvements for the Date/Time component. Read that post as an introduction to what you should and shouldn't do when working with dates and times in WordPress.

WordPress and timezones

One of the most important things to understand with dates in WordPress is that WordPress always works with `UTC` as a default timezone. You shouldn't try to change the default timezone with `date_default_timezone_set()`.

Timezones in WordPress are handled by the `timezone_string` setting in the database. WordPress calculates timezone offsets from that timezone setting.

To get the timezone with the setting from the database, you'll have two functions at hand:

1. **`wp_timezone()`** – Gets the site time zone as a `DateTimeZone` object.
2. **`wp_timezone_string()`** – Gets the site time zone as a string. Might return a `Region/Location` string or a `±NN:NN` offset.

Creating dates from strings

If you want to create a `DateTime` object or a timestamp from a time string, you have multiple possibilities. What you need to use depends on how you stored your dates and how you use them.

WordPress recommends to **store your dates either as Unix timestamps or formats that are precise moments in time**, such as `DATE_RFC3339` / `DATE_ATOM`.

Create a date from a timestamp

When you have a **timestamp**, you can create your object with `DateTimeImmutable` or `date_create_immutable()`.

```
$datetime = new DateTimeImmutable( '@' . $timestamp );
// or
$datetime = date_create_immutable( '@' . $timestamp )

// Note that we're reassigning here, since PHP's immutable functions/methods return new
values.
$datetime = $datetime->setTimezone( wp_timezone() );
```

Using `DateTimeImmutable` instead of `DateTime` is recommended by WordPress, because it's more predictable when working with different timezones.

It's important to set the timezone after you created the datetime object. You can't pass the timezone as the second parameter in `DateTimeImmutable::__construct()` or `date_create_immutable()`, because it will be ignored when you use a timestamp (See note in the documentation for `$timezone` parameter).

Create a date from a date string

When you know the format of the string, use `DateTimeImmutable::createFromFormat()` or `date_create_immutable_from_format()`.

```
$datetime = DateTimeImmutable::createFromFormat(
    DATE_ATOM,
    '2020-01-02T00:09:30+02:00'
);

// or

$datetime = date_create_immutable_from_format(
    DATE_ATOM,
    '2020-01-02T00:09:30+02:00'
);

$timestamp = $datetime->getTimestamp();
```

When the date string already includes the timezone, like when you use the `DATE_ATOM` format, then you don't need to pass a timezone. When it doesn't, you may have to pass it, depending on how you manage/use your dates.

If you stored your dates *with* a certain timezone applied, then you will have to create them with a timezone. You can do this by passing `wp_timezone()` as the third parameter.

```
$datetime = DateTimeImmutable::createFromFormat(
    'Y-m-d H:i',
    '2016-10-31 09:30',
    wp_timezone()
);

// or
```

```
$datetime = date_create_immutable_from_format(
    'Y-m-d H:i',
    '2016-10-31 09:30',
    wp_timezone()
);

// Note that we're reassigning here, since PHP's immutable functions/methods return new
values.
$datetime = $datetime->setTimezone(wp_timezone());

$timestamp = $datetime->getTimestamp();
```

Dates without times

The principle above also applies to dates without times: If you stored your dates with a certain timezone applied, then you will have to create them with a timezone.

A time zone **is still relevant** if you only need dates and not times. Because dates are created with the current time applied by default, time zone matters. Why? Here's an example: If your date object is created from the string `2016-10-31` at `00:30` at night and your timezone offset to UTC is `+1`, then you might end up with a date that's one day later, `2016-11-01`.

Here's the same example explained with code:

Don't do this

```
// Current time is 00:30, timezone is UTC + 1
$datetime = date_create_immutable_from_format('Y-m-d', '2016-10-31');

// 2016-10-31 23:30
echo $datetime->format('Y-m-d H:i');

// 2016-11-01 00:30
echo wp_date('Y-m-d H:i', $datetime->getTimestamp());
```

To work around that, **use** `wp_timezone()` when creating your datetime object.

Do this

```
// Current time is 00:30, timezone is UTC + 1
$datetime = date_create_immutable_from_format(
```

```
    'Y-m-d',
    '2016-10-31',
    wp_timezone()
);

// 2016-10-31 00:30
echo $datetime->format('Y-m-d H:i');

// 2016-10-31 00:30
echo wp_date('Y-m-d H:i', $datetime->getTimestamp());
```

strtotime()

If you don't know the exact format of the date, you can try using `strtotime()` or `date_create_immutable()`. Valid formats are explained in Supported Date and Time Formats.

```
$timestamp = strtotime('2008-08-07 18:11:31');

// No timezone needed, because it's already included in the string.
$datetime = date_create_immutable('2020-01-02T00:09:30+02:00');

// Either with or without a timezone, depending on how you saved your dates.
$datetime = date_create_immutable('2008-08-07 18:11:31', wp_timezone());
$datetime = date_create_immutable('2008-08-07 18:11:31');

// Either with or without a timezone, depending on how you saved your dates.
$datetime = new DateTimeImmutable('2008-08-07 18:11:31', wp_timezone());
$datetime = new DateTimeImmutable('2008-08-07 18:11:31');
```

Control the date display format

When you've worked with dates and times in PHP before, you're probably used to the `date()` function, or `DateTime::format()`. In WordPress, we usually don't use these function to change the date format. Instead, we used the `date_i18n()` function to get a **date in a translated format, using the correct timezone**. As of WordPress 5.3, there's the `wp_date()` function, which you should use whenever you can. It's a replacement for `date_i18n()`.

By default, Timber uses the date format set in *Settings → General*. That settings is saved in the `date_format` option.

```
// With a timestamp.
wp_date('F j, Y @ g:i a', $timestamp);

// With a DateTime object.
wp_date('F j, Y @ g:i a', $datetime->getTimestamp());
```

If you want to display a date in a different timezone than the site's timezone, use the `$timezone` parameter in `wp_date()`.

```
wp_date('F j, Y @ g:i a', $timestamp, 'Australia/Sydney');
```

Post dates

The date a post was published is accessible through `{{ post.date }}`.

```
{# With default date format from Settings → General #}
{{ post.date }}
```

Similarly, to get the date a post was modified, you can use `{{ post.modified_date }}`.

```
{# With default date format from Settings → General #}
{{ post.modified_date }}
```

If you want to change the display format, use an argument for the function. Check the documentation for date() to see which formatting options you can use.

Twig

```
{{ post.date('F j, Y @ g:i a') }}
{{ post.modified_date('F j, Y @ g:i a') }}
```

Twig filters and functions

Twig includes a `date` filter as well as a `date()` function. Timber supports this functionality out of the box and sets the correct timezones in the background. You don't have to set timezones in the *Twig Environment* yourself.

Remember, you should set the correct timezone in *Settings → General* in the WordPress admin to make this work correctly.

```
{{ my_date|date('j. F Y') }}
{{ post.date|date('j. F Y') }}
{{ post.modified_date|date('j. F Y') }}
{{ '2020-02-20 20:20'|date('j. F Y') }}
{{ 'now'|date('j. F Y') }}
```

Current date

To get the current date in WordPress, you can use one of the following functions:

- `time()` – Gets the current time as a timestamp.
- `current_datetime()` – Gets the current time as a `DateTimeImmutable` object in the site's timezone.
- `wp_date()` – Gets a formatted date with correct translation in the site's timezone.

Don't use the `date()` function in PHP to get the current date in a custom format. And remember, if you create new `DateTime` objects directly, they will be in the `UTC` timezone, and not in the timezone set in in your WordPress settings.

PHP

```
$timestamp = time();

$datetime_object = current_datetime();
$formatted_date = $datetime_object->format('Ymd');

$formatted_date = wp_date('Ymd');

// Don't do this.
$today = date('Ymd');
```

In Twig, you'll have more options with the `date()` function or the `now` keyword. Yes, while you shouldn't use `date()` in PHP, you can use it in Twig.

Twig

```
{# Current date as a DateTime object #}
{{ date() }}

{# Current date, formatted #}
```

```
{{ 'now'|date('F j, Y @ g:i a') }}
```

Time differences

WordPress comes with a handy `human_time_diff()` function, which returns the difference between two times in a human readable format, e.g. "1 hour", "5 mins", "2 days".

In Timber, you can use the `Timber\DateTimeHelper::time_ago()` function. The function also exists as a `time_ago` filter in Twig.

PHP

```
DateTimeHelper::time_ago($post->date());
```

Twig

```
{{ post.date('U')|time_ago }}
```

HTML

```
3 days ago
```

It works both for future and past dates.

Comparing dates

When you want to compare dates, then compare Unix timestamps, `DateTimeInterface` objects (like the DateTime class), or string–comparable dates in the same timezone.

```
$same = $timestamp === $timestamp;
$same = new DateTimeImmutable() === new DateTimeImmutable();
$same = wp_date('U') === time();

// Check if post publishing date is before today.
$before_today = $post->date('Ymd') < wp_date('Ymd');
$before_today = $post->date('U') < current_datetime()->getTimestamp();
```

In Twig, there's the `date()` function which you can use to compare dates.

```
{% if date(post.meta('show_until')) >= date('now') %}
    {# do something #}
{% endif %}
```

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