# Django-Tailwind

***Release 2.0.0***

**Tim Kamanin**

**Jan 13, 2022**

# CONTENTS

CHAPTER

ONE

# DEFINITIONS

This document uses the *Tailwind* word when we talk about two things: the CSS framework and the Django package.

So let's agree that we'll use:

- **Django Tailwind**, when we talk about this very package;
- **Tailwind CSS**, when we talk about the CSS framework;

So **Django Tailwind** was created to make **Tailwind CSS** and Django play together ().

## 1.1 Contents

### 1.1.1 Installation

#### Step-by-step instructions

1. Install the `django-tailwind` package via `pip`:

   ```
   python -m pip install django-tailwind
   ```

   Alternatively, you can install the latest development version via:

   ```
   python -m pip install git+https://github.com/timonweb/django-tailwind.git
   ```

2. Add `'tailwind'` to `INSTALLED_APPS` in `settings.py`:

   ```
   INSTALLED_APPS = [
     # other Django apps
     'tailwind',
   ]
   ```

3. Create a *Tailwind CSS* compatible *Django* app, I like to call it `theme`:

   ```
   python manage.py tailwind init
   ```

   > During the initialization step, you'll be asked to choose between **Just in time (`jit`)** and **Ahead of time (`aot`)** modes. While the `jit` mode is new and somewhat experimental in *Tailwind CSS*, I suggest choosing it for the best development experience. You can change the mode later with a simple configuration update. Check the *jit vs aot* section for more information.

4. Add your newly created `'theme'` app to `INSTALLED_APPS` in `settings.py`:

```
INSTALLED_APPS = [
  # other Django apps
  'tailwind',
  'theme'
]
```

5. Register the generated `'theme'` app by adding the following line to `settings.py` file:

```
TAILWIND_APP_NAME = 'theme'
```

6. Make sure that the `INTERNAL_IPS` list is present in the `settings.py` file and contains the `127.0.0.1` ip address:

```
INTERNAL_IPS = [
    "127.0.0.1",
]
```

7. Install *Tailwind CSS* dependencies, by running the following command:

```
python manage.py tailwind install
```

8. The *Django Tailwind* comes with a simple `base.html` template located at `your_tailwind_app_name/templates/base.html`. You can always extend or delete it if you already have a layout.

9. If you are not using the `base.html` template that comes with *Django Tailwind*, add `{% tailwind_css %}` to the `base.html` template:

```
{% load tailwind_tags %}
...
<head>
   ...
   {% tailwind_css %}
   ...
</head>
```

   The `{% tailwind_css %}` tag loads appropriate stylesheets and, when you're in `DEBUG` mode, connects to the `browser-sync` service that enables hot reloading of assets and pages.

10. Ok, now you should be able to use *Tailwind CSS* classes in HTML. Start the development server by running the following command in your terminal:

```
python manage.py tailwind start
```

Check out the *Usage* section for information about the production mode.

### Optional configurations

### Purge rules configuration

Depending on your project structure, you might need to configure the `purge` rules in `tailwind.config.js`. This file is in the `static_src` folder of the theme app created by `python manage.py tailwind init {APP_NAME}`.

For example, your `theme/static_src/tailwind.config.js` file might look like this:

```
module.exports = {
  purge: [
    // Templates within theme app (e.g. base.html)
    '../templates/**/*.html',
    // Templates in other apps
    '../../templates/**/*.html',
    // Ignore files in node_modules
    '!../../**/node_modules',
    // Include JavaScript files that might contain Tailwind CSS classes
    '../../**/*.js',
    // Include Python files that might contain Tailwind CSS classes
    '../../**/*.py'
  ],
  ...
}
```

Note that you may need to adjust those paths to suit your specific project layout. It is crucial to make sure that *all* HTML files (or files containing HTML content, such as `.vue` or `.jsx` files) are covered by the `purge` rule.

For more information about setting `purge`, check out the *"Controlling File Size"* page of the Tailwind CSS docs: https://tailwindcss.com/docs/controlling-file-size/#removing-unused-css - particularly the *"Removing Unused CSS"* section, although the entire page is a useful reference.

Under the **Ahead of time** (`aot`) mode, PurgeCSS only runs when you use the `python manage.py tailwind build` management command (creates a production CSS build).

If you run *Tailwind CSS* in the **Just in time** (`jit`) mode, you will get an optimized build even in development mode, and it happens at lightning speed.

See the *JIT vs AOT* section for more information about *Tailwind CSS* compilation modes.

### Configuration of the path to the `npm` executable

*Tailwind CSS* requires *Node.js* to be installed on your machine. *Node.js* is a *JavaScript* runtime that allows you to run *JavaScript* code outside the browser. Most (if not all) of the current frontend tools depend on *Node.js*.

If you don't have *Node.js* installed on your machine, please follow installation instructions from the official Node.js page.

Sometimes (especially on *Windows*), the *Python* executable cannot find the `npm` binary installed on your system. In this case, you need to set the path to the `npm` executable in *settings.py* file manually (*Linux/Mac*):

```
NPM_BIN_PATH = '/usr/local/bin/npm'
```

On *Windows* it might look like this:

```
NPM_BIN_PATH = r"C:\Program Files\nodejs\npm.cmd"
```

Please note that the path to the `npm` executable may be different on your system. To get the `npm` path on your system, try running the command `which npm` in your terminal.

### 1.1.2 Usage

#### Running in development mode

To start Django Tailwind in development mode, run the following command in a terminal:

```
python manage.py tailwind start
```

This will start a long-running process that watches files for changes. Use a combination of CTRL + C to terminate the process.

Several things are happening behind the scenes at that moment:

1. When *Tailwind CSS* `jit` mode is enabled, a stylesheet is updated every time you add or remove a CSS class in a Django template.
2. The `browser-sync` service watches for changes in HTML and CSS files. When a Django template page is updated, a browser reloads it. When a CSS file is updated, `browser-sync` applies updates without reloading the page. That gives you a smooth development experience without the need to reload the page to see updates.
3. The `nodemon` watches for config file changes (`tailwind.config.js`, `postcss.config.js`, `bs.config.js`) and restarts the long-polling process every time there's a change in those files.

If by somewhat reason you don't want to use hot-reloading, you can run the long-polling process with the `--no-sync` option to disable hot reloading:

```
python manage.py tailwind start --no-sync
```

#### Building for production

To create a production build of your theme, run:

```
python manage.py tailwind build
```

This will replace the development build with a bundle optimized for production. No further actions are necessary; you can deploy!

### 1.1.3 Settings

*Django Tailwind* comes with preconfigured settings. You can override them in the `settings.py` of your Django project.

##### TAILWIND_APP_NAME

This defines the *Tailwind* theme Django app containing your *Tailwind CSS* styles. I prefer to name such an app `'theme'`. You should generate the app during the installation phase by running the following command:

```
python manage.py tailwind init
```

Please refer to the *Installation* section for more information on the installation process.

#### TAILWIND_DEV_MODE

Determines whether the `browser-sync` snippet is added to the page via the `{% tailwind_css %}` tag. It is set to `DEBUG` by default, but you can override this value.

#### NPM_BIN_PATH

This defines a path to the `npm` executable in your system.

> *Tailwind CSS* requires you to have *Node.js* installed on your machine. *Node.js* is a *JavaScript* runtime that allows running *JavaScript* code outside a browser. Most of the current frontend tools depend on *Node.js*.
>
> If you don't have *Node.js* installed on your machine, please follow installation instructions from the official Node.js page.

The default value is:

```
NPM_BIN_PATH = 'npm'
```

#### TAILWIND_CSS_PATH

This defines a path to the generated *Tailwind CSS* stylesheet. If you have created a theme app via the `python manage.py tailwind init` command, chances are you don't need to change this value.

However, if you integrated *Tailwind CSS* in another way or want to use a *CDN* version of the bundle, you might want to change the path.

The default value is:

```
TAILWIND_CSS_PATH = 'css/dist/styles.css'
```

### 1.1.4 mode: Just in time VS Ahead of time

*Tailwind CSS* comes with two stylesheet generation modes to choose from:

1. Just in time (`jit`) - is the recommended mode. In this mode, Tailwind builds your stylesheet dynamically as you add/remove classes from your templates.
2. Ahead of time (`aot`) - is a legacy mode. In this mode, Tailwind builds the stylesheet with all classes available in the framework. As a result, you might end up with a huge stylesheet. To alleviate this, the production mode uses *Purge CSS* that removes unused styles from the stylesheet and dramatically reduces its size. But still, it's not as good as the `jit` mode.

If you run the `python manage.py tailwind init` command, you'll see a prompt to choose one of the modes. You can always change the selected mode later by editing your generated `theme/static_src/tailwind.config.js` file.

### 1.1.5 Template tags

*Django Tailwind* introduces a couple of template tags for your convenience.

#### {% tailwind_css %} tag

#### Usage

The `{% tailwind_css %}` tag generates a stylesheet link for the `'theme'` app and that's all you need to include *Tailwind's CSS* on a page:

```
{% load tailwind_tags %}
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Django Tailwind</title>
    {% tailwind_css %}
  </head>
  <body></body>
</html>
```

#### Asset versioning

The tag also supports asset versioning via the `v=` parameter, like this:

```
{% tailwind_css v='1' %}
```

Depending on your production setup, you may or may not need this functionality, so it's optional.

#### Hot reloading

This tag also includes the `browser-sync` script that reloads a page whenever the template changes. This script is only included on the page when two conditions are met:

- The debug mode is enabled in `settings.py`: `DEBUG=True`;
- Your IP address matches `INTERNAL_IPS` defined in `settings.py`. For local development, it usually should contain `127.0.0.1` to work properly:

  ```
  INTERNAL_IPS = ['127.0.0.1']
  ```

#### {% tailwind_preload_css %} tag

The tag generates a preload directive for your stylesheet, which improves loading performance in production. Place it above the `{% tailwind_css %}` tag:

```
{% load tailwind_tags %}
<!DOCTYPE html>
<html lang="en">
  <head>
```

(continues on next page)

(continued from previous page)

```
    <title>Django Tailwind</title>
    {% tailwind_preload_css %}
    {% tailwind_css %}
  </head>
  <body></body>
</html>
```

It also supports asset versioning (if needed):

```
{% tailwind_preload_css v='1' %}
```

### 1.1.6 Upgrading from Tailwind CSS 2.1 to 2.2

*Tailwind CSS* `2.2` has introduced a few breaking changes which means you have to update your `TAILWIND_APP` configuration.

In the following steps, we assume that your `TAILWIND_APP_NAME` is `theme`; replace it with your theme name if it is different.

1. Install the latest *Django-Tailwind* version via `pip`:

   ```
   pip install django-tailwind
   ```

2. Open the `theme/static_src/postcss.config.js` and remove the following two lines from it:

   ```
       tailwindcss: {},
       autoprefixer: {},
   ```

   We do this because `autoprefixer` is now built into *Tailwind CSS* `v2.2`, and `tailwindcss` is now a command-line tool, not a *postcss* plugin.

3. Now open the `theme/static_src/package.json` and update its `scripts` section to look like:

   ```
   "scripts": {
       "start": "npm run dev",
       "build": "npm run build:clean && npm run build:tailwind",
       "build:clean": "rimraf ../static/css/dist",
       "build:tailwind": "cross-env NODE_ENV=production tailwindcss --postcss -i ./
   ↪src/styles.css -o ../static/css/dist/styles.css --minify",
       "sync": "browser-sync start --config bs.config.js",
       "dev:tailwind": "cross-env NODE_ENV=development tailwindcss --postcss -i ./src/
   ↪styles.css -o ../static/css/dist/styles.css -w",
       "dev:sync": "run-p dev:tailwind sync",
       "dev": "nodemon -x \"npm run dev:sync\" -w tailwind.config.js -w postcss.
   ↪config.js -w bs.config.js -e js",
       "tailwindcss": "node ./node_modules/tailwindcss/lib/cli.js"
   },
   ```

   Here we replaced the `postcss` command with the `tailwindcss` command.

4. Staying in `theme/static_src/package.json`, find the *tailwindcss* dependency in `devDependencies` and make sure its value is set to `"~2.2.4"`. This way, you bind the *Tailwind CSS* version to the `2.2.x` branch, which gives you better control over future updates to the *Tailwind CSS* version.

5. Go to the `theme/static_src` directory and run the following command to install the updates:

```
npm install
```

Congrats, you're done with the upgrade!

### 1.1.7 Migrating from Django-Tailwind v1 to v2

> Please note that the instructions below are for upgrading the Django package, not for the actual dependency on Tailwind CSS.

*Django Tailwind2* `v2` introduces many new features that aren't available to projects generated with the previous version of the package. Thus if you want to get all the goodies `v2` offers, you need to update your Django `theme` app.

Depending on how many customizations you have, the process can be smooth or bumpy.

#### Steps to upgrade

Let's assume you've been using *Django Tailwind* for a while, and your `TAILWIND_APP_NAME` is `theme`.

1. Edit `INSTALLED_APPS` in `settings.py` file and remove the `'theme'` app.
2. Rename the `theme` app directory to `theme-legacy`.
3. Generate a new *Tailwind* theme app:

   ```
   python manage.py tailwind init
   ```

   Name it the same as your previous app: `theme`.
4. Add the `'theme'` back to `INSTALLED_APPS`;
5. Make sure that `INTERNAL_IPS` list is present in the `settings.py` file and contains the `127.0.0.1` ip address:

   ```
   ```python
   INTERNAL_IPS = [
       "127.0.0.1",
   ]
   ```
   ```
6. Copy `theme-legacy/static_src/src` to `theme/static_src/src`;
7. If you have a file named `theme/static_src/src/styles.scss`, rename it to `theme/static_src/src/styles.css`. In `v2` we dropped support for *SASS*, but *POSTCSS* should work just fine. Unless you've used advanced *SASS* features, which is unlikely;
8. Open `theme-legacy/static_src/tailwind.config.js` and compare it with `theme/static_src/tailwind.config.js`, if you have customizations there, like custom colors, variables, etc., copy them to `theme/static_src/tailwind.config.js`;
9. Notice the `plugins` listed in `theme/static_src/tailwind.config.js`. We've now included the four official *Tailwind CSS* plugins there. If you see that your forms look weird after the update, you probably don't need the official `@tailwindcss/forms` package, so disable it by removing the following line:

   ```
   require('@tailwindcss/forms'),
   ```

   from the `theme/static_src/tailwind.config.js`.
10. Copy `theme-legacy/templates` to `theme/templates`;

11. Open `theme/templates/base.html` and add `{% load tailwind_tags %}` to the beginning of the file. Then, replace:

```
<link
  rel="stylesheet"
  href="{% static 'css/styles.css' %}"
  type="text/css"
/>
```

with

```
{% tailwind_css %}
```

12. To install dependencies, run the following command:

```
python manage.py tailwind install
```

13. Now start the development server:

```
python manage.py tailwind start
```

14. If all went well, you should now be on the latest version of *Django Tailwind* with your previous styles intact.

### 1.1.8 Updating *Tailwind CSS* and its dependencies

When a new release of *Tailwind CSS* comes out, you can update your `theme` project without updating *Django Tailwind*.

#### Checking if there are updates for *Tailwind CSS* and its dependencies

Before doing an update, you can check if there are any updates. Run the following command:

```
python manage.py tailwind check-updates
```

This command runs the `npm outdated` command behind the scenes in the context of your `theme/static_src` directory.

If there are updates, you'll see a table of dependencies with the latest compatible versions. If there are no updates, this command will return no output.

#### Updating Tailwind CSS and its dependencies

If you want to use the latest version of *Tailwind CSS*, run the following command:

```
python manage.py tailwind update
```

This command runs the `npm update` command behind the scenes in the context of your `theme/static_src` directory.

If there are updates, you'll see a log of updated dependencies. If there are no updates, this command will return no output.

### 1.1.9 Running in Docker

You can find a Docker example under the `example` directory.

Check the included `example/Dockerfile` and `example/docker-compose.yml` for more information.

Here's how to start the `example` project via Docker:

1. Go into the `example` directory;

```
cd example
```

2. Build containers via `docker-compose`:

```
docker-compose build
```

3. Start containers:

```
docker-compose up
```

4. Open `http://localhost:8000` in a browser. You should see the main page.