# Installation

## Step-by-step instructions

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

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

   If you want to use automatic page reloads during development (see steps 10-12 below) use the `[reload]` extras, which installs the `django-browser-reload` package in addition:

```
python -m pip install 'django-tailwind[reload]'
```

   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
```

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

The `{% tailwind_css %}` tag includes Tailwind's stylesheet.

10. Let's also add and configure `django_browser_reload`, which takes care of automatic css refreshes in the development mode. Add it to `INSTALLED_APPS` in `settings.py`:

latest

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

11. Staying in `settings.py`, add the middleware:

```
MIDDLEWARE = [
  # ...
  "django_browser_reload.middleware.BrowserReloadMiddleware",
  # ...
]
```

The middleware should be listed after any that encode the response, such as Django's `GZipMiddleware`. The middleware automatically inserts the required script tag on HTML responses before `</body>` when `DEBUG` is `True.`

12. Include `django_browser_reload` URL in your root `url.py`:

```
from django.urls import include, path
urlpatterns = [
    ...,
    path("__reload__/", include("django_browser_reload.urls")),
]
```

13. Finally, 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

## Content (formerly Purge) rules configuration

The `content` section of your `tailwind.config.js` file is where you configure the paths to all of your HTML templates, JavaScript components, and any other source files that contai

latest

class names.

Depending on your project structure, you might need to configure the `content` 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 = {
    content: [
        // 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 `content` rule.

For more information about setting `content`, check out the *"Content Configuration"* page of the Tailwind CSS docs: https://tailwindcss.com/docs/content-configuration.

## 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'
```

latest

On *Windows*, you may have npm on `$PATH` but it's `npm.cmd` rather than `npm`. (You can call it from the terminal because `$PATHEXT` contains `.cmd`.) If so, please override the default `NPM_BIN_PATH = 'npm'`:

```
NPM_BIN_PATH = 'npm.cmd'
```

Alternatively (and for maximum reliability), you can use a fully qualified path. 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 for your system. To get the `npm` path, try running the command `which npm` in your terminal. (On *Windows*, please try `where npm` or `Get-Command npm`)

If you share codes with others, you can search `$PATH` (and `$PATHEXT` on Windows) dynamically in *settings.py*:

```
from shutil import which
NPM_BIN_PATH = which("npm")
```