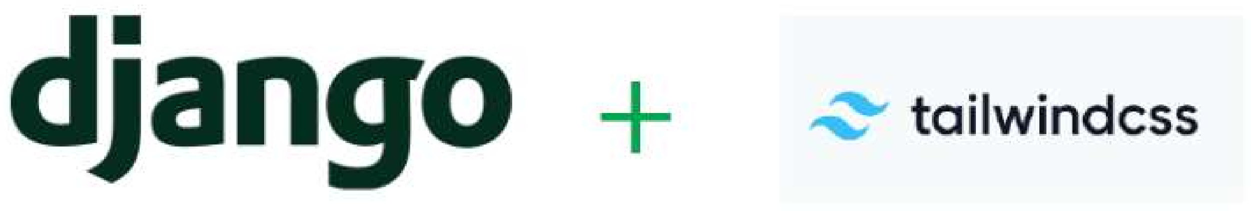
Taiwo Jazz
Posted on Sep 7, 2023 • Updated on Sep 9, 2023

❤️ 15 🦄 1 🙌 1

Setting up Tailwindcss with Django: Easy Guide



In this tutorial, we will walk through the process of setting up Tailwind CSS to work globally for all apps within your Django project. Assuming you have already created your Django project and app, let's dive straight into it.

Step 1: Installing Tailwind CSS

Begin by navigating to your project's root directory, where your manage.py file resides. Execute the following commands to install Tailwind CSS, along with its necessary dependencies, PostCSS, and Autoprefixer:

```
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init
```

These commands will install TailwindCSS, PostCSS, Autoprefixer and generate a `tailwind.config.js` default configuration file.

Step 2: Configuring TailwindCSS

You should now have a `tailwind.config.js` file in your project's root directory. Replace its contents with the following code:

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    './app-name/templates/app-name/**/*.html',
    // Add paths to other apps if necessary
  ],
  theme: {
    extend: {},
  },
  plugins: [],
};
```

Make sure to replace `app-name` with your actual app name. If your project includes multiple apps, add the respective paths to their templates.

Step 3: Configuring Postcss

In your project's root directory, create a new file named `postcss.config.js` and paste the following code inside it:

```
module.exports = {
  plugins: {
    tailwindcss: {},
    autoprefixer: {},
  }
}
```

Step 4: Add the Tailwind directives to your CSS

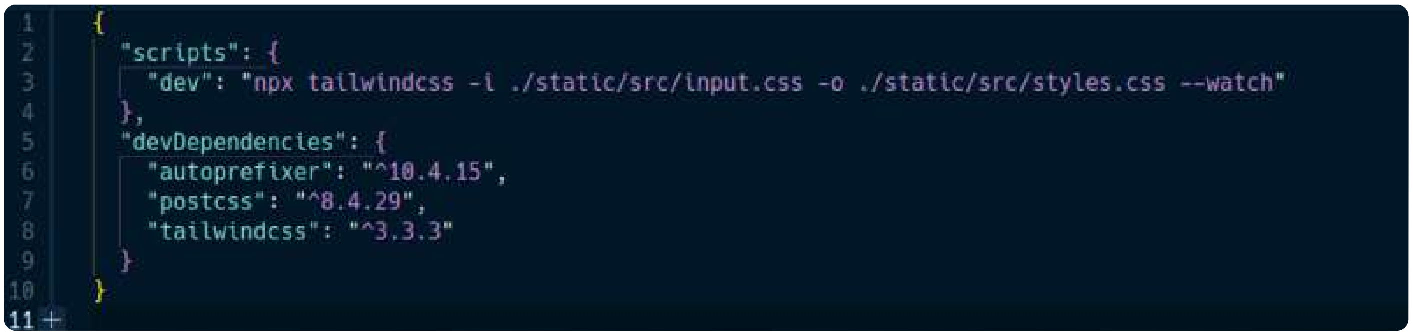
Create a directory named `static` in your root directory. Inside this directory, create another directory named `src`, and within it, create a file named `input.css`. Add the following code to `input.css`:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Step 5: Update your package.json

Locate your `package.json` file in the root directory, and add the following line at the top of the `devDependencies` object:

```
"scripts": {
    "dev": "npx tailwindcss -i ./static/src/input.css -o ./static/src/styles.css --watc
  },
```

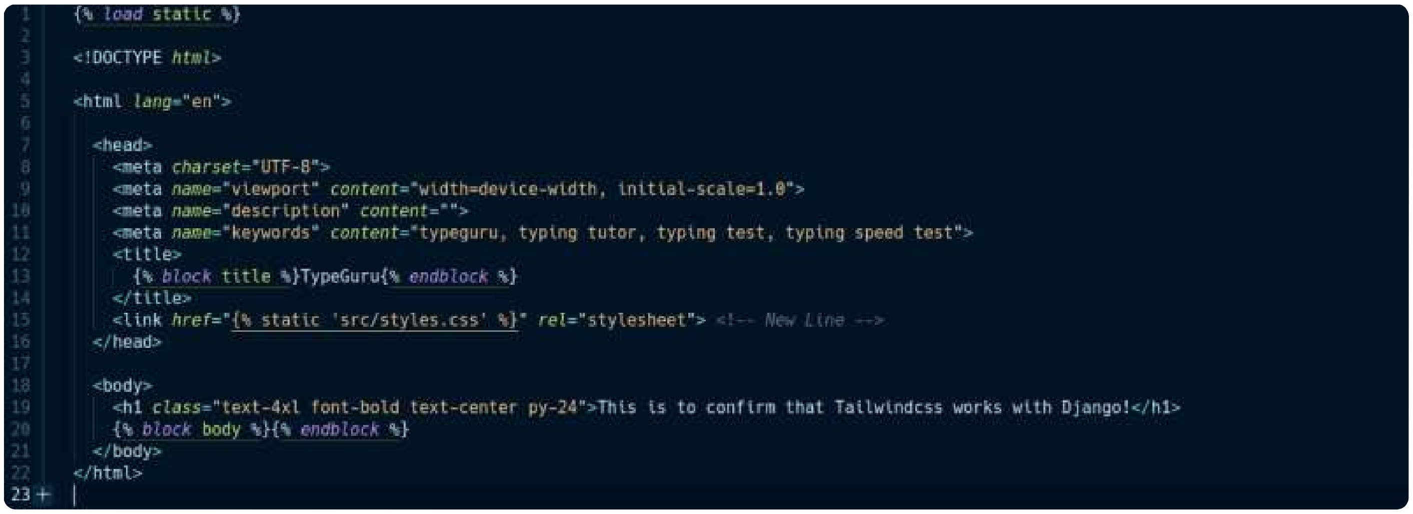


```
{
  "scripts": {
    "dev": "npx tailwindcss -i ./static/src/input.css -o ./static/src/styles.css --watch"
  },
  "devDependencies": {
    "autoprefixer": "^10.4.15",
    "postcss": "^8.4.29",
    "tailwindcss": "^3.3.3"
  }
}
```

Step 6: Update your app templates layout.html or base.html

In your app's templates directory, where your `.html` files are located, update the head section of your `layout.html` or `base.html` file with the following code. This ensures that your stylesheet points to the CSS files generated by Tailwind when you start the server:

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



```
{% load static %}

<!DOCTYPE html>

<html lang="en">

  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
    <meta name="keywords" content="typeguru, typing tutor, typing test, typing speed test">
    <title>
      {% block title %}TypeGuru{% endblock %}
    </title>
    <link href="{% static 'src/styles.css' %}" rel="stylesheet"> <!-- New Line -->
  </head>

  <body>
    <h1 class="text-4xl font-bold text-center py-24">This is to confirm that Tailwindcss works with Django!</h1>
    {% block body %}{% endblock %}
  </body>
</html>
```

Step 7: Update Django settings

In your `settings.py` file, follow these steps:

1. Import the `os` Module

Add the following code at the top of your settings.py file, typically around line 13:

```
import os
```

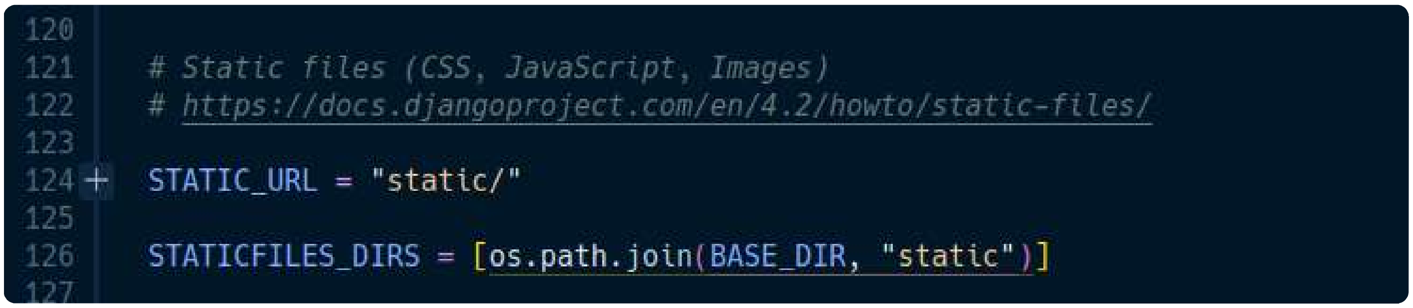
This import is necessary to work with file paths.

2. Configure Static Files Directory

Locate the `STATIC_URL` declaration in your settings.py file (usually at the bottom of the page), and then add the following code right below it:

```
STATICFILES_DIRS = [os.path.join(BASE_DIR, "static")]
```

Your `settings.py` file should now include the updated `STATICFILES_DIRS` configuration, ensuring that Django knows where to find your `static` files.



```

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC_URL = "static/"

STATICFILES_DIRS = [os.path.join(BASE_DIR, "static")]

```

With these changes, your Django project is configured to recognize the `static` directory as a source of static files. This is crucial for serving the CSS generated by Tailwind CSS. Save your `settings.py` file, and you're all set.

Step 8: Now open a new terminal aside the one running your django server and run:

```
npm run dev
```

With these steps completed, you have successfully set up Tailwind CSS to work globally for all apps in your Django project. Enjoy creating stylish and responsive user interfaces for your Django applications.

Implementing Auto-Reload in Django

If you find manual page reloading tedious while working on your Django project, you can set up automatic page reloads whenever your server code, templates, content, or classes change. Here's how to do it:

Step 1: Install `django-browser-reload`

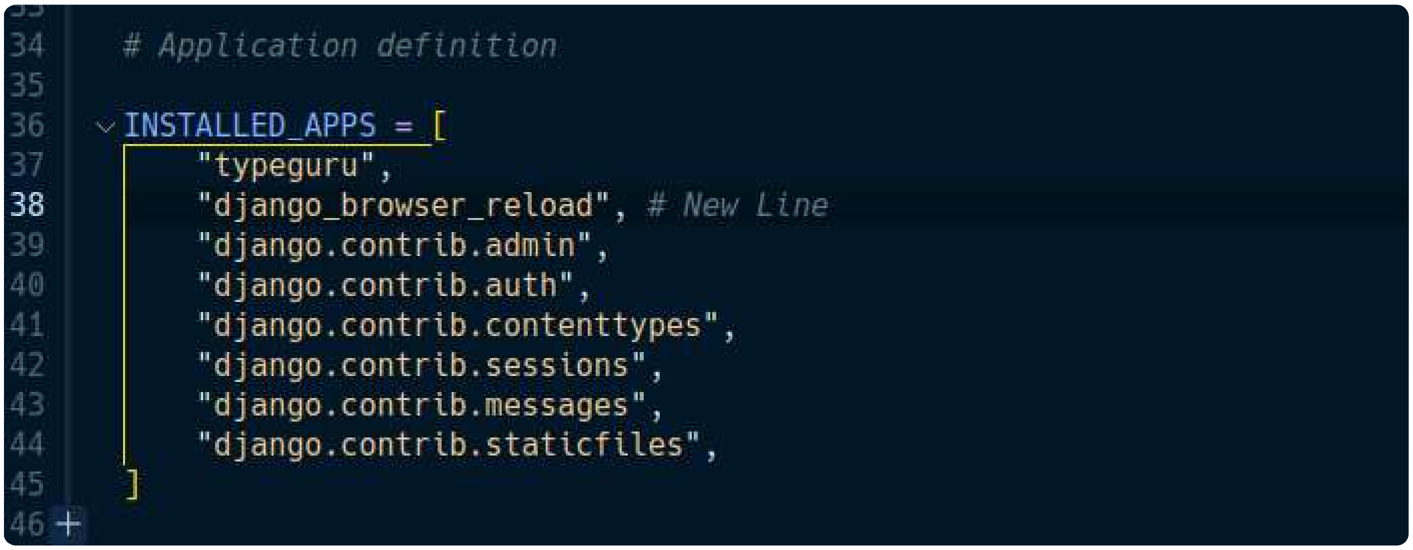
Start by installing `django-browser-reload` using `pip`:

```
python -m pip install django-browser-reload
```

Step 2: Add `django-browser-reload` to your `INSTALLED_APPS`:

In your Django project's settings, locate the `INSTALLED_APPS` list and add `django_browser_reload` to it:

```
INSTALLED_APPS = [
    ...,
    "django_browser_reload",
    ...,
]
```



```
# Application definition

INSTALLED_APPS = [
    "typeguru",
    "django_browser_reload", # New Line
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
]
```

Step 3: Include the App URLs

Extend your project's URL configuration to include the `django-browser-reload` app's URLs. In your project's main `urls.py` (usually located in the project's root directory), add the following:

```
from django.urls import include, path

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

This step ensures that the auto-reloading functionality is accessible at the `__reload__/` endpoint of your Django project.

Step 4: Add the middleware:

In your project's settings, locate the `MIDDLEWARE` list and add `django_browser_reload.middleware.BrowserReloadMiddleware` to it. Make sure to place it after any other middleware that encodes the response, such as Django's `GZipMiddleware`.

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

With these steps completed, your Django project is now set up to automatically reload the page whenever there are changes in your server code, templates, content, or classes.

Enjoy a more streamlined development experience with automatic page reloading!

👋 Before you go ...

Please leave your appreciation by commenting on this post!

It takes ***one minute*** and is worth it for your career.

Get started

Top comments (0)

Code of Conduct • Report abuse

Mailgun PROMOTED

Build with the platform that enables you to send, track, and optimize emails effortlessly

Over 150,000 companies are building great apps and email programs with Mailgun. See what you can accomplish with the world's best email delivery platform.

Get Started Now

Taiwo Jazz

I'm a very ambitious software engineer and self-taught front-end developer that is passionate about bringing business logic and designs to life.

LOCATION
Nigeria

JOINED
Nov 6, 2022

More from Taiwo Jazz

How to setup Django + Postgres + Node.js with Docker
#django #postgres #docker #node

Setting Up CS50 Library Locally in Visual Studio Code
#computerscience #cs50 #c #programming

How to Run Multiple Instances of Google Chrome in KDE Neon Plasma
#linux #kdeplasma #chrome

AWS PROMOTED ...

Get a wealth of knowledge from AWS data experts

Dig in on data questions with expert advice

Let’s Talk About Data is an information gold mine for data professionals looking for the latest news and guidance on AWS Data & Analytics services and partners.

Learn More