
Django Learning Report

Name : Prajwal Bashyal
Institution: Bhaktapur Multiple Campus, BIT
Date : 2025-12-20
Topic : Django – Static Files complete guide

Django Static Files – Short Written Steps (Quick Revision)

1. Static files are **CSS, JavaScript, images, fonts** used in a website.
2. Create a **static** folder at project level and keep CSS, JS, and images inside it.
3. In `settings.py`, define `STATIC_URL = '/static/'`.
4. Add static directory path using: STATIC_DIR = BASE_DIR/'static'
5. `STATICFILES_DIRS = [STATIC_DIR]`.
6. Django searches static files first in `STATICFILES_DIRS`, then in app static folders.
7. In templates, load static using `{% load static %}`.
8. Access static files using `{% static 'path/to/file' %}`.
9. Images automatically get URLs using `STATIC_URL`; no separate configuration is needed.

Django Static Files – Complete Guide

This section explains **static files in Django using app-based structure**, which is the **recommended and professional approach** for scalable projects.

---

1. What Are Static Files?

Static files are files that do **not change dynamically**:

- CSS files
- JavaScript files
- Images
- Fonts

They are used to style and enhance HTML pages.

---

2. Why Use App-Level Static Files?

Using app-level static files:

- Avoids filename conflicts between apps
- Keeps each app self-contained
- Is used in **real-world and reusable Django apps**
- Works best for large projects (blog, accounts, dashboard, etc.)

---

3. App-Level Static Folder Structure

Each app contains its **own static folder**, and **app name is mandatory** inside it.

Example (`myfirstapp`):

```
project_root/
├── myfirstapp/
│   ├── static/
│   │   └── myfirstapp/
│   │       ├── css/
│   │       │   └── style.css
│   │       ├── js/
│   │       │   └── main.js
│   │       └── images/
│   │           └── logo.png
│   ├── templates/
│   │   └── myfirstapp/
│   │       ├── base.html
│   │       └── home.html
│   ├── views.py
│   └── models.py
```

📌 **The app name inside `static/` is required**.

---

4. Static Configuration in `settings.py`

For app-level static files, **no extra directory is needed**.

```
STATIC_URL = '/static/'
```

Django automatically searches:

- `myfirstapp/static/myfirstapp/`

As long as the app is in `INSTALLED_APPS`.

---

5. Django Static Files Search Flow

When Django sees:

```
{% static 'myfirstapp/css/style.css' %}
```

Django searches:

1. `myfirstapp/static/myfirstapp/css/style.css`
2. Other apps' static folders
3. `STATIC_ROOT` (production)

If not found → **404 error**

---

6. Using Static Files in Templates

Step 1: Load static tag

```
{% load static %}
```

Step 2: Link CSS

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

Step 3: Link JavaScript

```
<script src="{% static 'myfirstapp/js/main.js' %}"></script>
```

Step 4: Link Images

```
<img src="{% static 'myfirstapp/images/logo.png' %}" alt="Logo">
```

📌 App name **must be included** in the path.

---

7. Image URL Logic

Every static image automatically gets a URL:

How Django generates URL

Django combines **STATIC_URL** with the **file path inside static folder**:

STATIC_URL= ‘static/’

```
STATIC_URL + app_name + file_path
```

Example: Resulting url :

```
/static/myfirstapp/images/logo.png
```

No manual URL configuration is required.

---

8. Templates vs Static (Clear Seperation)

Folder	Purpose
`templates/`	HTML files only
`static/`	CSS, JS, images

✕ Do not put CSS or images inside templates
✕ Do not mix static files with HTML

---

9. Common Mistakes

- Forgetting app name in static path
- Forgetting `{% load static %}`
- Wrong folder nesting