

# CHAPTER 1: Introduction to Web Development: Frontend Basics, Web Frameworks, and Django History

---




## What is Frontend?

**Frontend** refers to the part of a website that users interact with directly.

It includes everything users see: text, images, buttons, forms, layout, and colors.

The frontend is built using three core technologies:

The frontend is built using three core technologies:

	Technology	Purpose
	HTML	Structure of the page
	CSS	Styling and layout
	JavaScript	Interactivity ( <i>skip for now</i> )

**HTML:** HTML, which stands for HyperText Markup Language, is the standard language used to create and structure content on the web. It is not a programming language, but a markup language that uses tags to define elements such as headings, paragraphs, links, images, and more. These tags help web browsers understand how to display the content. For example, `<p>` defines a paragraph, while `<h1>` defines a heading. Every webpage you see is built on a structure made with HTML. It acts like the skeleton of a webpage, giving shape and meaning to all the visible content. HTML files are written in plain text and saved with the .html extension. It is essential for anyone who wants to build websites, and it forms the base upon which more advanced technologies like CSS and JavaScript work.

**CSS:** CSS, which stands for Cascading Style Sheets, is the language used to style and design the appearance of HTML elements on a webpage. While HTML structures the content (like text, headings, images), CSS is what makes that content look good—it controls the layout, colors, fonts, spacing, alignment, and even animations. For example, if HTML creates a button, CSS decides its background color, border, size, and hover effect. CSS can be written directly inside HTML, but it's usually kept in a separate .css file to keep the code clean and organized. It works by selecting HTML elements (like all `<p>` tags or elements with a specific class) and applying styling rules to them. CSS makes websites attractive, responsive, and user-friendly across different screen sizes and devices. Without CSS, all web pages would look plain and boring—just black text on a white background.

**JS:** JavaScript (JS) is a programming language used to make web pages interactive and dynamic. While HTML gives a web page its structure and CSS gives it style, JavaScript brings it to life. With JS, you can add features like buttons that respond when clicked, forms that check for errors, sliders, pop-ups, animations, and much more. It can also fetch data from a server without reloading the whole page—this is how modern web apps like Gmail or Facebook work smoothly. JavaScript is run directly in the browser, and it can be written inside the HTML file or in a separate .js file. It can access and change HTML and CSS through something called the DOM (Document Object Model). JavaScript is one of the core technologies of web development and is essential for creating responsive, user-friendly websites and applications.

## Why are Web Frameworks Important?

1. **Speed up development:** They provide ready-made components, so you don't waste time reinventing the wheel.
2. **Keep code organized:** Frameworks enforce patterns and structure that make code easier to read, maintain, and scale.
3. **Security:** Many frameworks have built-in protections against common web vulnerabilities like SQL injection or cross-site scripting.
4. **Community support:** Popular frameworks have large communities, plugins, and tutorials that make learning and problem-solving easier.
5. **Best practices:** Frameworks encourage or enforce industry best practices, so you write better, cleaner code.

### Examples of Web Frameworks

- **Django (Python):** A powerful, “batteries-included” framework that comes with tools for everything—database handling, authentication, admin panels, and more. Great for building complex, secure websites quickly.
- **Flask (Python):** A lightweight, minimal framework that gives you more control. It's great for smaller apps or when you want to add only what you need.
- **React (JavaScript):** Although technically a frontend library, it's often called a framework because it helps build user interfaces efficiently with reusable components.
- **Express.js (JavaScript/Node.js):** A fast, minimalist backend framework for building APIs and web servers with Node.js.
- **Ruby on Rails (Ruby):** A convention-over-configuration framework focused on rapid development with clean, readable code.

# django

## History of Django

Django is a high-level Python web framework that was originally created to make building complex, database-driven websites faster and easier. It was developed in 2003 by two web developers, Adrian Holovaty and Simon Willison, who were working at a local newspaper called the Lawrence Journal-World in Lawrence, Kansas, USA.

At that time, they faced the challenge of quickly developing and maintaining several newsroom websites with tight deadlines. To solve this, they built Django as an internal framework to handle repetitive web development tasks like database access, URL routing, template rendering, and user authentication.

Because it worked so well, they decided to release Django as an open-source project in July 2005. Since then, Django has grown massively in popularity due to its “batteries-included” philosophy—meaning it comes with almost everything you need to build web apps out of the box.

Over the years, Django has been continuously improved by a large global community, becoming one of the most popular and reliable frameworks for Python web development.

Got it, Bisham! Here's a polished exercise and Q&A section you can include in your chapter on “Introduction to Web Development: Frontend Basics, Web Frameworks, and Django ”History”—designed to be clear, practical, and engaging for learners.

## Section A: Short Answer Questions

1. What is the role of frontend in web development?
2. Define a web framework.
3. Name any two frontend technologies.
4. What are the basic technologies used to build a frontend interface?
5. Mention one major difference between frontend and backend.
6. When was Django first released?
7. Who developed Django?
8. What language is Django written in?
9. Mention one reason Django became popular after its release.
10. Name any two popular web frameworks apart from Django.

## Section B: Long Answer Questions

11. Explain the basic structure and technologies of the frontend in web development.
12. Describe what a web framework is and why it is used.
13. Compare and contrast frontend and backend development.
14. Write a detailed note on the history of Django.
15. Discuss the evolution of Django and its contribution to modern web development.

## Exercise & Questions

Exercise: Build a Simple Static Webpage

Objective:

Create a static webpage using HTML, CSS, and a little JavaScript that introduces the History of Django and explains what a Web Framework is.

### Instructions:

- **HTML: Structure your page with the following sections:**

- A main heading with the title “History of Django and Web Frameworks”
- A paragraph summarizing the history of Django (use simple, clear sentences)
- Another paragraph explaining what a web framework is and why it’s important

16. **CSS:** Style your webpage to make it visually appealing. For example:

- Choose a pleasant font family and size
- Add colors to headings and paragraphs
- Use spacing (padding/margin) to separate sections
- Add a border or background color to the page container

- 

17. **JavaScript:** Add a simple interactive feature such as:

- A button that shows or hides the Django history paragraph when clicked
- Or a button that changes the background color of the page on click

18. **Save your files:**

- Create index.html for your HTML
- Create styles.css for your CSS (link it properly in HTML)

Bonus:

Try adding an image related to Django or web development and center it on the page.

Deliverables:

- Submit your HTML, CSS, and JS files
- Make sure your code is clean and well-indented
- Test your webpage in a browser before submitting

# CHAPTER 2: Installation of Python and Django and MVC

---

## Download and Install Python:

### Step 1:

Go to <https://www.python.org/downloads/> and download latest python.

Python is an interpreted, high-level programming language runtime that executes Python code. The Python installation includes the interpreter (python.exe), standard library, and tools needed to run .py files on your system.

### Step 2:

Install python : During installation, check ☒ “Add Python to PATH”

Adding Python to the system PATH during installation allows you to run Python and its tools (like python, pip, etc.) directly from the command line without needing to navigate to the Python installation folder. The PATH is an environment variable that tells the operating system where to look for executable programs. Without adding Python to PATH, you’d have to type the full path to the Python executable every time you want to use it, which is inconvenient and error-prone. Adding it makes Python accessible globally from any directory in the terminal, simplifying development and automation tasks.

### Step 3:

Open Command Prompt and run: `python --version`  
You should see something like Python 3.12.x

### Step 4:

Install pip (Python package manager)  
Usually comes with Python.  
Check with:

pip is the package installer for Python. It allows you to download, install, and manage third-party Python libraries and tools from the Python Package Index (PyPI) or other sources.

### Step 5:

Create a virtual Environment:  
`python -m venv <name of virtual enviroment>`  
eg: `python -m venv venv`

Activate Virtual Environment:  
`venv/Scripts/activate`

A virtual environment in Python is an isolated workspace that allows you to keep the dependencies (packages and libraries) required for a project separate from those used by other projects or the system-wide Python installation. This is important because different projects may require different versions of the same package, and installing everything globally can lead to conflicts and errors. By creating a virtual environment, you ensure that each project has its own clean, self-contained setup, making your development process more organized, reliable, and easier to manage. It’s a best practice followed by professionals to avoid dependency issues and maintain project stability.



- ◆ `urls.py`
  - The router of your project.
  - Maps URLs to views using `urlpatterns`.
- ◆ `asgi.py` and `wsgi.py`
  - These files are entry points for deployment.
  - `wsgi.py` is used for most production deployments (Apache, Gunicorn).
  - `asgi.py` is used for asynchronous servers like Daphne (for websockets, etc).

## Adding an App

A Django app is a modular component of a Django project. It's like a self-contained feature or unit of your website – blog, users, payments, etc.

A Django project can contain many apps. A Django app can be reused in multiple projects.

Why Create Django Apps?

- Organizes code by feature or functionality
- Makes it easier to manage, test, and reuse
- Follows Django's "pluggable apps" philosophy

How to Create a Django App

Inside your Django project, run: `python manage.py startapp <app name>`

example : `python manage.py startapp blog`

This creates a folder `blog/` with core files.

## File Structure of a Django App

`__init__.py`

- Makes `blog` a Python package
- Usually left empty

`admin.py`

- Used to register models for the Django admin interface.

`apps.py`

- Django's internal config for the app.
- Class-based config:

`models.py`

- Define your database schema (tables)

`views.py`

- Defines how the app responds to requests.

`tests.py`

- For writing unit tests for your app.

`urls.py` (you create this manually)

- Defines routes for this app:

```
blog/
├── __init__.py
├── admin.py
├── apps.py
├── models.py
├── views.py
├── tests.py
├── urls.py      <-- (you create this manually)
└── migrations/
    └── __init__.py
```

`migrations/`

- Stores database change files generated from your models.

## What is MVC?

MVC stands for Model View Controller.

It's a software design pattern that separates application logic into three distinct parts, making your code modular, manageable, and scalable.

Used widely in web development, desktop apps, and even mobile app.

## Purpose of MVC.

Keep data, logic, and UI separate.

Make the code easier to maintain, test, and scale.

Enable team collaboration – frontend and backend developers can work independently.

## The MVC Components (with Explanation)

### Model

Manages the data and business logic.

- Deals with the database or other data sources.
- Defines the structure, rules, and relationships of your data.

### View

Handles the presentation/UI.

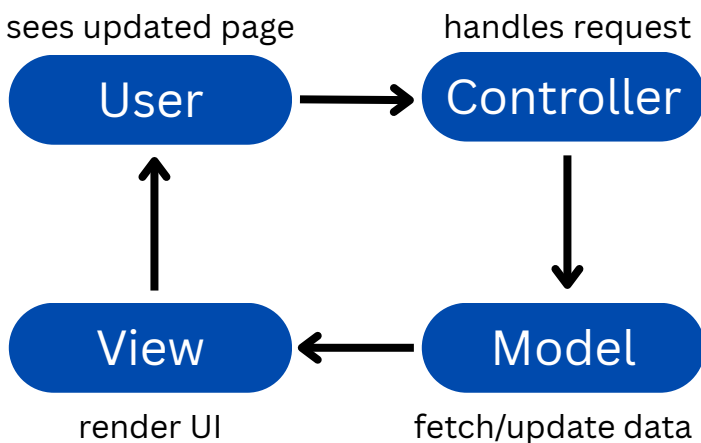
- What the user sees.
- Displays data from the Model.
- Gets data via the Controller.

### Controller

Handles user input and routes it to the right logic.

- Processes requests and returns responses.
- Connects the View and the Model.

## Flow of MVC



## Benefits of MVC

- Separation of Concerns: UI and business logic are isolated.
- Modular Code: Easier to maintain, test, and reuse.
- Collaborative: Backend devs can work on models/controllers, frontend devs on views.
- Scalability: Easily add features or shift UI without touching backend logic.

Django provides a command-line interface (CLI) through the `manage.py` file. These commands let you:

- Create apps
- Run development server
- Handle database migrations
- Create superusers
- Interact with the database
- Run custom scripts

### 1. `python manage.py runserver` – Start Development Server

`python manage.py runserver`

Starts Django's built-in development server.

### 2. `startapp` – Create a New App

`python manage.py startapp blog`

Creates a new Django app folder with required files.

### 3. `makemigrations` – Create Migration Files

`python manage.py makemigrations`

Detects changes in `models.py` and creates migration files (like versioned DB instructions).

### 4. `migrate` – Apply Migrations to the Database

`python manage.py migrate`

Applies all unapplied migrations to your actual database.

### 5. `createsuperuser` – Create Admin User

`python manage.py createsuperuser`

Creates an admin login for the Django admin dashboard.

### 6. `shell` – Open Django Python Shell

`python manage.py shell`

Starts a Python shell with full access to your Django models and settings.

### 7. `showmigrations` – List Migrations

`python manage.py showmigrations`

Shows which migrations exist and whether they've been applied.



## Introduction

PostgreSQL is a powerful, open-source object-relational database system. Django supports PostgreSQL as one of its official database backends. Using PostgreSQL with Django provides advanced features such as JSON fields, full-text search, and better performance for large-scale applications.

## Prerequisites

Before setting up PostgreSQL with Django, ensure the following are installed:

- Python
- pip
- Django
- PostgreSQL
- psycopg2 (PostgreSQL adapter for Python)

## Step-by-Step Guide to Configure PostgreSQL with Django

### Step 1: Install PostgreSQL

Download and install PostgreSQL from the official site:

During installation, note down:

- Username (default: postgres)
- Password
- Port (default: 5432)

### Step 2: Create a PostgreSQL Database

Use pgAdmin or the command line to create a new database.

- Example command (on terminal or psql shell):
- CREATE DATABASE myprojectdb;
- You can also create a new user (optional):
- CREATE USER myuser WITH PASSWORD 'mypassword';
- GRANT ALL PRIVILEGES ON DATABASE myprojectdb TO myuser;

### Step 3: Install psycopg2

- Inside your Django project virtual environment, install the PostgreSQL adapter:
- pip install psycopg2-binary
- Step 4: Configure settings.py
- In your Django project folder, open the settings.py file and locate the DATABASES section. Replace the default SQLite configuration with PostgreSQL configuration:

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.postgresql',  
        'NAME': 'myprojectdb',  
        'USER': 'myuser',  
        'PASSWORD': 'mypassword',  
        'HOST': 'localhost',  
        'PORT': '5432',  
    }  
}
```

### Step 5: Apply Migrations

Once the configuration is complete, apply initial migrations to set up the default database schema:

```
python manage.py migrate
```

This will create all the necessary tables required by Django's built-in apps.

### Step 6: Run the Server

To verify that everything works correctly, run the development server:

```
python manage.py runserver
```

If the server runs without error, your database configuration is complete.

### Choose the correct option for each question.

1. What does MVC stand for in software architecture?

- a) Model-View-Code
- b) Model-View-Component
- c) Model-View-Controller
- d) Method-Variable-Controller

2. In Django, which component is responsible for managing the data and business logic?

- a) View
- b) Controller
- c) Middleware
- d) Model

3. Which file is used to run server commands in Django?

- a) server.py
- b) settings.py
- c) manage.py
- d) app.py

4. What is the default port Django's development server runs on?

- a) 3000
- b) 5000
- c) 8080
- d) 8000

5. Which command is used to start a new Django project?

- a) django startapp
- b) python create\_project
- c) django-admin startproject
- d) startproject django-admin

### Section B: Short Answer Questions

1. Define the role of the Model in the MVC pattern.
2. What is the function of the manage.py file in a Django project?
3. Write the command to install Django using pip.
4. Mention any two Django server commands with their purpose.
5. List the basic components created when starting a Django project.

### Section C: Long Answer Questions

6. Explain the MVC design pattern in detail with the role of each component.
7. Describe the steps required to install Python and Django on a Windows system.
8. Explain at least five Django management commands with examples.
9. Discuss how Django implements the MVC pattern using its own terminology (MTV).

### Section D: Practical Project

Project Title: Create a Basic Django App Demonstrating MVC Pattern

Objective:

To install Django, set up a project, run the server, and create an app that implements the MVC (MTV) design pattern.