**Choosing Between FastAPI and Flask**

When deciding between **FastAPI** and **Flask** for building web APIs or backend services in Python, FastAPI is often the better choice for modern applications. This document outlines the key differences and why FastAPI is generally preferred.

**Why FastAPI is Better**

|  |  |  |
| --- | --- | --- |
| **Feature** | **FastAPI** | **Flask** |
| **Performance** | High performance using async and Starlette | Lower performance (sync-based) |
| **Type Safety** | Full support with Python type hints & Pydantic | Manual validation or third-party libs |
| **Automatic Docs** | Built-in Swagger UI & ReDoc | Requires plugins (e.g., Flask-RESTx) |
| **Async Support** | Native support for async/await | Limited or needs extensions |
| **Modern Python** | Designed for Python 3.6+ | Older design, no enforced typing |
| **Validation** | Automatic request validation | Manual or plugin-based |
| **Dependency Injection** | Lightweight built-in DI system | Not included |

**Use FastAPI If You:**

* Need high performance and async capabilities
* Want automatic documentation (OpenAPI)
* Prefer strong typing and auto-validation
* Are building modern microservices or APIs

**Use Flask If You:**

* Are working on a legacy Flask project
* Need a very minimal and lightweight app
* Don't require async or automatic validation

**Conclusion**

**FastAPI** is the superior choice for most modern use cases involving APIs, offering speed, simplicity, and powerful features out of the box. Choose **Flask** mainly if you need minimalism or are continuing an older project.