FastHTML

FastAPI + opinionated web development tools

What?

For those that just want to build stuff, FastHTML focuses on a “one size fits all” opinionated approach to building web applications. This opinionated approach (aka “the Django approach”) is in contrast to total freedom of design (aka “the Flask approach”).

Why?

> “Any Flask/FastAPI HTML-focused application of enough complexity eventually becomes a home grown analogue of Django.”

FastAPI is awesome, but…

FastAPI is awesome, but it is optimized for API development, not HTML-focused efforts. This isn’t a criticism of FastAPI, rather a compliment to the focus of the framework. FastHTML will provide tooling on top of FastAPI to enhance and empower web development and do so in a well-documented and opinionated manner.

Opinionated Design

The opinion is important because a lot of developers want components handed to them so they can assemble them into projects, rather than figure out fundamentals. In other words, the wheel they want to invent isn't in the Python domain, rather their business product.

Planned Features

1. HTMX first, with support for HTMX headers
2. FastHTML-specific Helpers for Jinja2 templates
3. Standardized authentication
4. "fh" as a standard prefix
5. Django-to-FastHTML migration instructions
6. SQLModel first!
7. Option to use Django models instead of SQLAlchemy
    a. Allows plugging in of Django admin
    b. Still works with FastAPI
8. Cookiecutter-powered template for base projects
9. Standardized architecture based on domain modeling rather than Django's component design
10. Standard user model
11. Classless CSS by default, use minimal CSS library

Why not just do it with Django?

1. Because we like FastAPI and pydantic and SQLModel ;-)
2. Because the Async story in Django still isn't awesome
3. Because Django isn't opinionated enough: FastHTML will make HTMX a default
4. Django types are not part of the framework, adding friction to any project using them

5. Because other frameworks bring in newer ideas of utility, for example Redwood JS's naming methodology makes file search much better than Django

How?

With a bunch of coding

When?

Probably January 2024, maybe for a rebuild of Fileridge or the original of the historian project

Funding

1. Tie to FastAPI corp
2. Put together book-quality public-facing HTML docs. For an extra fee you can have them in other formats (epub, PDF, etc). Would need to create a private bridge between mkdocs and jupyter book

Questions

1. Has someone already built this?
2. Do I want to maintain this as a public framework?
3. If I do this as a public framework, is funding a potential option?