ChatGPT

FountainAI Comprehensive Tutorial Series (Beginner to Expert)

Overview: This structured series of tutorials takes a leveled approach to learning the FountainAI ecosystem. Each tutorial builds on the previous ones, gradually increasing complexity from **beginner** to **expert**. The series covers all core components – **Teatro** (UI DSL), **FountainStore** (persistence), **MIDI2** (multimedia), and the **FountainAI core services** – while emphasizing modern Swift development via the terminal (using SwiftPM, **no Xcode required** [1]). Every tutorial uses the same foundational tech stack for consistency [2], focusing only on essential, evidence-backed concepts (no fluff). Each tutorial targets a specific audience and learning goal, ensuring an optimal learning experience for **all types of users** from newcomers to advanced developers.

Tutorial 1: *Hello FountainAI – Setting Up Your First Project* (Beginner)

Target Audience: Developers new to FountainAI (and even to Swift) who want to start from scratch without complexity.

Goal: Set up the development environment and create a minimal FountainAI app using the template scaffold, all via the command line.

- **Environment Setup:** Install Swift 6.1+ and required tools (SwiftPM) on macOS 14+ [3]. Verify you can run Swift commands in Terminal (no Xcode needed).
- **Scaffolding a New App:** Use FountainAI's provided scaffold script to generate a starter macOS app template [4]. For example, cloning the **fountainai** monorepo and running `Scripts/new-gui-app.sh` will create a ready-made project structure [5].
- **Build & Run from Terminal:** Compile the app with Swift Package Manager and run it via Terminal (using `swift build` and `swift run/open` commands [6]). Ensure the default template app launches successfully.
- **Template Tour:** Briefly explore the generated project files. Identify where the **Teatro** UI, **FountainStore** integration, etc., are included in the template (scaffold includes all core dependencies by default [7]).
- **Outcome:** A "Hello, FountainAI" app running on your Mac, confirming your setup is correct. You'll have confidence in using Terminal-based Swift development and understand the starting project structure.

Tutorial 2: *Building a Basic UI with Teatro* (Beginner)

Target Audience: Beginners who have a running scaffolded app and want to learn how to create user interfaces using FountainAI's UI framework (Teatro).

Goal: Learn to build a simple user interface with **Teatro's declarative DSL** and SwiftUI, using the template as a base.

- **Intro to Teatro:** Understand what Teatro is – a SwiftUI-compatible declarative view engine with its own DSL for building interfaces and timeline animations [2] . No prior SwiftUI knowledge beyond basics is required.
- **Layout a Simple Interface:** Modify the scaffolded app's UI to display custom content. For example, create a simple screen with a title, some text, and a button using Teatro's DSL syntax (which gets converted to SwiftUI under the hood).
- **Interactive Elements:** Implement a basic interaction (e.g., a button click that updates a label or prints to console) to see how event handling works in Teatro.
- **Run & Verify:** Build and run the app from Terminal to test your new UI. The app should now show your custom interface and respond to user interaction.
- **Outcome:** You can confidently use Teatro to define UI components and layout. This sets the stage for more complex interfaces later. *(By reusing Teatro across tutorials, you reinforce a consistent UI approach [8] .)*

Tutorial 3: *Data Persistence with FountainStore – Saving and Loading Data* (Intermediate)

Target Audience: Developers with basic UI working, ready to add data storage to their apps. Ideal for those who want their app to remember data between runs.

Goal: Introduce **FountainStore**, FountainAI's embedded ACID database, and integrate it for saving/loading persistent data in the app.

- **What is FountainStore:** Learn how FountainStore serves as the "memory core" of FountainAI – a local, Swift-native storage engine that is **safe (crash-resistant), fast, and private (no external servers)** [9] .
- **Integrate the Store:** Use the FountainStore API to save user-generated data. For example, extend the app to have a simple note-taking feature: when a user enters text and hits save, it's stored in FountainStore.
- **Retrieve and Display Data:** On app launch (or via a "Load" action), query FountainStore for saved notes and display them in the UI (perhaps in a list). Demonstrate retrieving by ID or key and show that data persists across restarts.
- **Data Model & Indexing:** Define a basic data model for your records. Optionally, show how FountainStore can handle indexed searches or versioning if relevant (e.g., retrieving records by a tag or keyword to illustrate its capabilities [10]).
- **Outcome:** Your app now retains information between runs, proving the persistence works. You've learned to **safely save and fetch data** using FountainStore's simple API, gaining confidence that nothing will be lost even on crashes [11] .

Tutorial 4: *Multimedia and Timing with MIDI2* (Intermediate)

Target Audience: Developers who have basic UI and data handling down, and want to enrich their apps with sound or other timed multimedia features.

Goal: Learn to use the **MIDI2** library to add audio playback or timed events, and coordinate these with the app's UI (potentially using Teatro's timeline capabilities).

- **Intro to MIDI2:** Understand that MIDI2 is not just for music keyboards – in FountainAI it's used to **coordinate audio tracks and multimedia playback** [2]. It handles precise timing and control of media.
- **Play a Sound or Music Track:** Integrate MIDI2 to play an audio file or MIDI sequence in response to an app event. For example, when a user clicks a "Play" button, use MIDI2 APIs to start playback of a short sound or background music.
- **Timeline Synchronization:** (Optional advanced step) Tie the playback to UI changes. For instance, use Teatro's timeline animation DSL alongside MIDI2 so that UI elements animate in sync with the audio. This shows how **Teatro's timeline** and MIDI2 can work together for rich multimedia experiences.
- **Controls and Stop/Start:** Add simple controls (play/pause buttons) and use MIDI2 to handle these actions. Demonstrate responding to MIDI2 callbacks or status (e.g., track ended).
- **Outcome:** Your app now can produce sound and handle timing-based events. You've gained experience in multimedia integration – a user clicking a button can trigger audio and even synchronized UI animations. This opens the door to creating interactive audiovisual apps, an important aspect of the FountainAI stack.

Tutorial 5: *Integrating FountainAI's Intelligence – AI and OpenAPI Services* (Advanced)

Target Audience: Advanced developers ready to leverage FountainAI's unique AI capabilities in their applications. Ideal for those interested in AI assistants, automation, or FountainAI's **OpenAPI**-exposed services.

Goal: Connect your app to the FountainAI's cognitive services and AI features – making the app "smart." This includes querying FountainAI's running services (via their OpenAPI endpoints) and possibly using an LLM for content generation or analysis.

- **FountainAI Core Services:** Brief overview of FountainAI's internal services (e.g., **Semantic Browser**, **Planner**, **Baseline Memory**, etc.) and their OpenAPI specifications. You'll learn what each service does (for example, Semantic Browser might retrieve knowledge, Planner might execute tasks).
- **Running FountainAI Backend:** Instructions to launch the FountainAI services (if not already running). This might involve running the FountainAI monorepo's server components so that your app can query them. (Ensure you have an OpenAI API key configured if needed for LLM functionality [3].)
- **Example Integration – Semantic Query:** In your app, perform an AI-driven action. For instance, query the **Semantic Browser service** to fetch some info or status and display it. *(The provided example "HelloFountainAITeatro" does this – it queries a running Semantic Browser and renders the response [12].)* You'll write code to call a FountainAI OpenAPI (likely via a generated client or HTTP request) and handle the response.
- **Example Integration – AI Response:** Alternatively or additionally, use FountainAI's LLM integration to generate content. For example, allow the user to ask a question in-app and get an AI-generated answer. This would demonstrate hooking into FountainAI's language model gateway or using an OpenAI API directly through FountainAI's tools.
- **Using the Planner:** (Optional advanced exercise) Demonstrate the **Planner service** by sending a multi-step goal from the app and letting FountainAI orchestrate actions, then showing the result

or plan outcome in the UI. This showcases how the app can leverage FountainAI's autonomous planning capability.
- **Outcome:** Your application can now converse with or utilize AI logic from FountainAI. It's "intelligent" – capable of retrieving knowledge or performing complex tasks via the FountainAI backend. Through this, you've touched on the **full range of OpenAPI-accessible services** (though not every service in depth), gaining insight into how FountainAI's unique communication and planning abilities can be embedded into a user-facing app.

Tutorial 6: *End-to-End Project – Building a Screenplay Editor* (Expert)

Target Audience: Experienced developers who have gone through prior tutorials (or equivalent experience) and are ready to build a full-featured application using **all aspects** of the FountainAI ecosystem. This is also a great reference for how everything comes together in a real project.

Goal: Create a complex application – a macOS **screenwriting editor** – from scratch using the template-first approach. This project will incorporate **UI, persistence, multimedia, and AI features all together**, serving as a capstone for the tutorial series.

- **Project Overview:** The tutorial guides you to build a screenplay editor that combines text editing for script writing with multimedia playback capabilities. It uses the Fountain screenplay format as an example domain.
- **Start from Template:** Scaffold the base project using FountainAI's GUI template (as in Tutorial 1) to quickly get the skeleton app with all dependencies ready [13] .
- **UI Development with Teatro:** Design a rich text editing interface for screenplays using Teatro. Include formatting, scene list views, or timeline views for scenes. The UI is built declaratively and can animate scene playback (using Teatro's timeline for scrolling or highlights).
- **Data Management with FountainStore:** Store screenplay content, versions, and metadata using FountainStore. The editor auto-saves drafts to the local store, ensuring **no work is lost (ACID guarantees)**, and allows searching through the script or past revisions efficiently.
- **Multimedia Integration:** Integrate **MIDI2** to synchronize an audio track or sound effects with the screenplay. For example, allow playing dialogue audio or background score in sync with the script timeline. The tutorial demonstrates how to **synchronize playback with the script's timeline** (e.g., turning script pages or highlighting lines in time with audio) [14] .
- **AI-Assisted Features:** Add intelligent features with FountainAI – for instance, a "Story Coach" that uses AI to suggest improvements or generate descriptive text. Leverage Codex/LLM integration to auto-complete dialogue or summarize scenes directly within the app [15] . This shows how developers can use AI to enhance user creativity.
- **From Scaffold to Customization:** As you progress, heavily customize the initially scaffolded code to fit the screenplay editor's needs. This includes modifying default UI, adding new model types, and extending functionality while still following the FountainAI architectural patterns (scaffold gives the starting point, you build the specifics on top of it).
- **Full Workflow:** By the end, you will have implemented the **complete development workflow** – scaffolding, building UI, adding persistence, integrating multimedia, using AI, and packaging the app. The tutorial reinforces all prior concepts in one project [15] , truly covering the **full reach of the FountainAI ecosystem** in a real-world app.
- **Outcome:** A working **Fountain-based Screenplay Editor** application. This capstone demonstrates how powerful applications can be created rapidly by combining FountainAI's template-driven development with its unique services. You've now mastered everything from basic setup to advanced customization, equipped to create your own FountainAI-powered apps or even contribute new ideas to the ecosystem.

Conclusion and Next Steps

By following this comprehensive tutorial series, learners progress from basic to expert-level proficiency in FountainAI development. They will have covered **every major component** of the Fountain Coach organization's domain – from **Teatro UI design** to **FountainStore persistence**, **MIDI2 multimedia coordination**, and the FountainAI core's **AI services** – in a gradual, structured manner. Importantly, each tutorial focused only on what's truly needed to achieve its goal, with real examples and outcomes (no unnecessary "nice-to-haves"), aligning with the template-first, evidence-based philosophy of FountainAI development [16].

With this foundation, developers will find the **FountainAI Tutor** repository the go-to guide for learning and referencing how to build with FountainAI. Whether you're a Swift beginner or seasoned developer, these tutorials ensure you gain clear understanding of FountainAI's nature, its unique communication style, and the full extent of its capabilities. Armed with this knowledge, you can confidently explore the FountainAI ecosystem further, implement your own projects, or even extend the platform – truly **making the FountainAI Coach your creative partner** in development.

---

[1] [2] [3] [4] [5] [6] [7] [8] [13] [14] [15] [16] README.md
https://github.com/Fountain-Coach/tutor/blob/6d9699a3d0e7fc12bb6df0a61040373ccd4f362f/README.md

[9] [10] [11] README.md
https://github.com/Fountain-Coach/Fountain-Store/blob/c182b976f872d11b2e89bcc5187776fb2dea0ce3/README.md

[12] README.md
https://github.com/Fountain-Coach/the-fountainai/blob/7b9e624cfbea218c70994e5dc55319c1a234c9f5/Examples/HelloFountainAITeatro/README.md