Fresher-Level FastAPI Questions & Answers

Q: What is FastAPI?
A: FastAPI is a modern, fast (high-performance) web framework for building APIs with Python 3.7+ based on standard Python type hints.

Q: Why use FastAPI over Flask or Django?
A: FastAPI is asynchronous, faster, auto-generates documentation, and provides better performance and developer experience out-of-the-box.

Q: How do you install FastAPI and Uvicorn?
A: Use pip: pip install fastapi uvicorn

Q: What are the key features of FastAPI?
A: Asynchronous support, data validation, automatic docs (Swagger/OpenAPI), high performance, and dependency injection.

Q: How do you define a basic GET endpoint in FastAPI?
A: By using @app.get("/path") decorator on a function in a FastAPI app.

Q: What is Uvicorn and why is it used with FastAPI?
A: Uvicorn is an ASGI server used to run asynchronous Python web apps like those built with FastAPI.

Q: How does FastAPI handle request and response?
A: It uses Pydantic models to parse request data and serialize response data.

Q: How do you run a FastAPI application?
A: Use the command: uvicorn main:app --reload

Q: How is Swagger UI integrated in FastAPI by default?
A: FastAPI auto-generates Swagger UI at /docs using the OpenAPI standard.

Q: What are path parameters and query parameters in FastAPI?
A: Path parameters are part of the URL path, query parameters come after ? in a URL.

Intermediate-Level FastAPI Questions & Answers

Q: What are Pydantic models and how are they used in FastAPI?
A: Pydantic models define and validate data shapes for request and response bodies using Python type hints.

Q: Explain request body handling with FastAPI.
A: Request bodies are parsed into Pydantic models automatically if specified in route handlers.

Q: How do you validate incoming JSON data in FastAPI?
A: Using Pydantic models which enforce types, constraints, and validations automatically.

Q: How do you use Path, Query, and Body parameters?
A: Using FastAPI's Path, Query, and Body classes to define metadata and validations.

Q: What is the role of Dependency Injection in FastAPI?
A: Depends allows injecting reusable logic (auth, DB session) into endpoints.

Q: How can you return custom status codes and headers?
A: Use JSONResponse, Response, or return tuples like (data, status_code).

Q: What are background tasks in FastAPI?
A: FastAPI can run background tasks after sending response using BackgroundTasks class.

Q: Explain how CORS is handled in FastAPI.

A: Use CORSMiddleware from starlette.middleware.cors to allow cross-origin requests.

Q: How do you serve static files in FastAPI?
A: Use StaticFiles from Starlette to serve static files.

Q: How to create and use middleware in FastAPI?
A: Define a function that wraps request handling and add with add_middleware().

Advanced-Level FastAPI Questions & Answers

Q: How does FastAPI handle asynchronous programming?
A: FastAPI is built on ASGI and supports async/await for concurrent request handling.

Q: What are the best practices for structuring a large FastAPI project?
A: Use routers, services, models, and config modules to keep code modular and maintainable.

Q: How to implement OAuth2 or JWT-based authentication in FastAPI?
A: Use OAuth2PasswordBearer, and Depends to secure endpoints with token validation.

Q: How do you test FastAPI applications?
A: Using FastAPI's TestClient or HTTPX with pytest for endpoint testing.

Q: Explain how FastAPI uses Starlette under the hood.
A: FastAPI is built on Starlette and inherits routing, middleware, and ASGI features.

Q: How do you integrate SQLAlchemy with FastAPI?
A: Use session management and Depends for DB session injection in endpoints.

Q: How can you achieve API versioning in FastAPI?
A: By prefixing routers with /v1, /v2, etc., and grouping endpoints by version.

Q: What are the limitations or disadvantages of FastAPI?
A: Newer ecosystem, async DB support still evolving, steep learning curve for beginners.

Q: Explain event handlers startup and shutdown in FastAPI.
A: Use @app.on_event("startup") and @app.on_event("shutdown") to run setup/cleanup logic.

Q: How would you implement rate limiting or throttling in FastAPI?
A: Use third-party middleware like slowapi or integrate with Redis for rate-limiting.