Speaker: A. Jesse Jiryu Davis

Title: What Is Async, How Does It Work, and When Should I Use It?

Abstract:

Storing state: Web applications typically receive client requests, do some I/O with a back end, like a database or API server, and finally respond to the client. The app must store state while waiting for the backend. In the case of websockets, the app waits arbitrarily long for a server-side event before responding to the client.

If the server uses threads to store that state, the programming model is easy to understand but has high costs per connection. The recent history of web development has trended towards higher connection counts, and larger amounts of per-connection state that applications must store.

Async frameworks use various techniques to store state without threads: asyncio uses coroutines and callbacks.. These techniques minimize resources per connection, but they're unfamiliar.

Look at an example chat server written in 20 lines using Flask. Compare to the same application with asyncio. Note that since Flask uses a thread per connection, state is stored implicitly in language facilities like the stack, the stack pointer, and the program counter. Describe how these concepts map to the asyncio example.

A structured tour of the asyncio event loop. We start at the APIs used by the example chat application: loop.start_serving() and loop.run_forever(). Examine their code (greatly simplified to be easy to read on slides). start_serving() sets a socket to be non-blocking, and adds its file descriptor to a list of descriptors on which it's waiting for events. It waits using Kqueue or Epoll or some other platform-specific event service. run_forever() begins the event loop itself. Explore the event loop's basic logic, and see how it supports concurrent operations in a single thread.

Look at bad example code that mixes synchronous operations with

asyncio. Based on our understanding of asyncio, we know that this has awful consequences: it blocks the event loop. With relief, see how to fix the bad example.

When to use async: Good for apps with high connection counts, especially in service-oriented architectures, or if server events are pushed to the client over WebSockets. Takes time to learn, longer to code, harder to debug. Async is a poor choice if you rely on synchronous third-party libraries like database drivers.