1. When to use callbacks

- For **simple synchronous computations**, a blocking call is simpler.

- For **CPU-intensive tasks in parallel**, sometimes **CompletableFuture with join()** or **parallel streams** is cleaner.

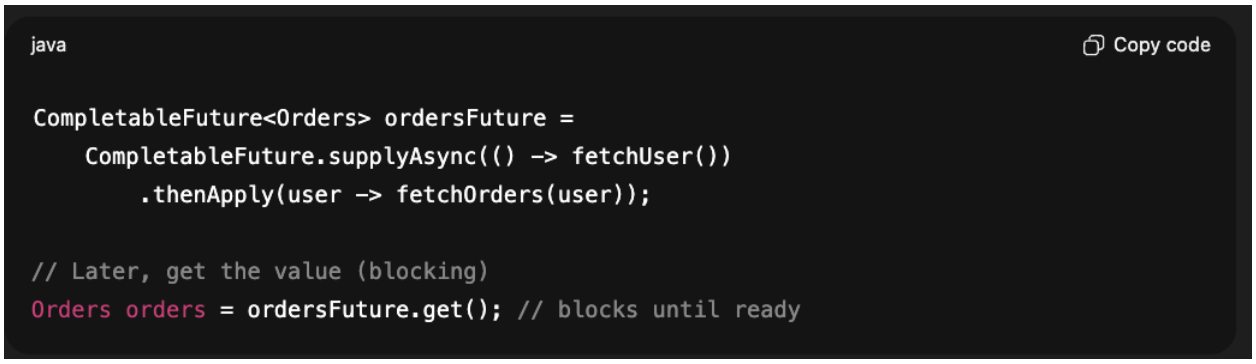
2. Future

You must manage the response blocking by yourself, do not support callback.

`get()`	Waits (blocks) until computation completes and returns result
`cancel()`	Cancel the computation
`isDone()`	Check if the computation is finished
`isCancelled()`	Check if it was cancelled

3. CompletableFuture

Have callback for long CPU-intensive task, manage thenApply (process), thenAccept (success), exception(Error) => Easy to manage chain.



```java
CompletableFuture<Orders> ordersFuture =
    CompletableFuture.supplyAsync(() -> fetchUser())
        .thenApply(user -> fetchOrders(user));

// Later, get the value (blocking)
Orders orders = ordersFuture.get(); // blocks until ready
```