Lập trình đồng bộ tại 1 thời điển chỉ có 1 việc được thực thi -> đang load dữ liệu thì không thể làm việc khác -> lập trình bất đồng bộ

Hàm async khi được gọi sẽ không block hàm khác nghĩa là

Khi hàm async chạy, gọi một hàm khác thì hàm khác đó được chạy song song với async ( hàm này có thể async hoặc không)

Khi đang gọi hàm async A, có thể gọi A thêm 1 lần nữa, khi đó ta có 2 A chạy song song nhau -> không tốt -> hàm async thường có await để đợi hàm đó thực thi xong

Await trong hàm async

+ đợi câu lệnh sau await thực hiện xong thì mới thực hiện những câu lệnh kế tiếp

Câu lệnh sau async là bất đồng bộ:

Nếu không có await thì trong thời gian câu lệnh đó thực thi, các câu lệnh kế tiếp sẽ chạy mà không chờ câu lệnh async đó xog

Await khi gọi hàm async:

Tương tự thì hàm async là một câu lệnh async, nên ko có await thì các câu lệnh tiếp theo sẽ chạy dù trong lúc đó hàm async chưa thực hiện xong, những câu lệnh này có thể phụ thuộc vào kết quả hàm async -> phải await để đợi hàm async xong

Hàm async thường có await, ít nhất là await complete task để khi nó chưa xong thì không gọi lại chính nó, bởi vì mỗi async sẽ được coi như 1 luồng bất đồng bộ, khi ko có await dẫn đến tạo ra nhiều luồng giống nhau -> làm phúc tạp vấn đề

Await trong hàm chỉ đợi dòng lệnh của hàm đó, nếu lúc gọi hàm ko có await thì cái thread khác ở cùng scope chổ gọi hàm sẽ chạy bình thường như ko có await, còn trong thread của hàm đang chạy vẫn await trong body

<https://www.c-sharpcorner.com/article/async-and-await-in-c-sharp/>