<https://www.youtube.com/watch?v=eV45ZgXU1Mk>

* What does problem async await solve? The problem of long running operations.
* Blocking: when the thread is busy doing stuff (or waiting), two types
  + CPU bound blocking operation – CPU thread of execution is busy because of computation
  + IO bound blocking operation – CPU has to wait on an external resource such as hard drive, folder, file system, network access. Execution is waiting, doing nothing, for a result.
* Impact of blocking operations, depends on app type
  + On client application, the app becomes unresponsive
  + On server application, requests can become queued and hog threads
* How do we solve this?
  + Option 1: write code that employs multiple threads aka multithreading. Very complex, hard to debug, most developers avoid this option.
  + Option 2: async, promises, futures.
    - Promise Model of Concurrency
      * Breaks the call of the async method apart from the value of the result we would expect to get back
* Example

Var message = await httpClient.GetAsync(someUrl); // returns the promise of a future value, once right hand side completes goes to next line

Private async Task<string> CallingAsync()

{

Var httpClient = new HttpClient();

Var message = await httpClient.GetAsync(“http://localhost/api/function1”);

Var content = await message.Content.ReadAsStringAsync();

Return content;

}

Private async void button\_click()

{

Var result = await CallingAsync();

Print(result);

}