- Example

An example of the functionality of std::async in the scope of concurrency in C++.

WE'LL COVER THE FOLLOWING ^

- Example
 - Explanation

Example

```
// asyncLazyEager.cpp
#include <chrono>
#include <future>
#include <iostream>
int main(){
  std::cout << std::endl;</pre>
  auto begin= std::chrono::system_clock::now();
  auto asyncLazy=std::async(std::launch::deferred,
                             []{ return std::chrono::system_clock::now(); });
  auto asyncEager=std::async(std::launch::async,
                              []{ return std::chrono::system_clock::now(); });
  std::this_thread::sleep_for(std::chrono::seconds(1));
  auto lazyStart= asyncLazy.get() - begin;
  auto eagerStart= asyncEager.get() - begin;
  auto lazyDuration= std::chrono::duration<double>(lazyStart).count();
  auto eagerDuration= std::chrono::duration<double>(eagerStart).count();
  std::cout << "asyncLazy evaluated after : " << lazyDuration</pre>
            << " seconds." << std::endl;</pre>
  std::cout << "asyncEager evaluated after: " << eagerDuration</pre>
            << " seconds." << std::endl;</pre>
  std::cout << std::endl;</pre>
```



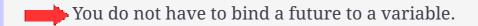




[]

Explanation

- Both std::async calls (at lines 13 and 16) return the current *time point*. The first call is lazy, while the second call is eager. The short pause of one second in line 19 demonstrates this fully.
- The call <code>asyncLazy.get()</code> (line 21) will trigger the execution of the promise in line 13 the result will be available after a short nap of one second (line 19). If you do not ask for the value with <code>asyncLazy.get()</code>, the promise would never run.
- This is not true for asyncEager.get() which gets the result from the immediately executed work package.



Let's test your understanding of this topic with an exercise in the next lesson.