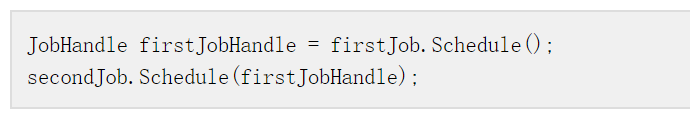
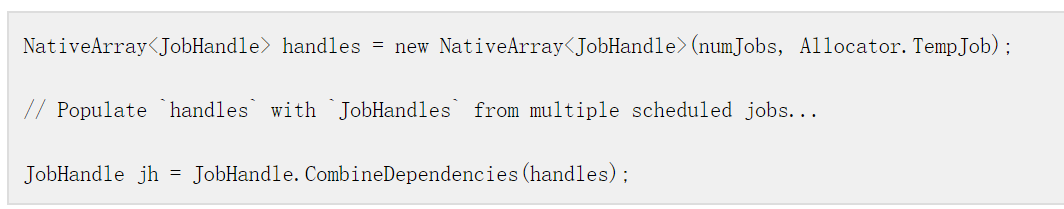
**JobHandle and dependencies**

When you call the [Schedule](https://docs.unity3d.com/ScriptReference/Unity.Jobs.IJobExtensions.Schedule.html) method of a job it returns a [JobHandle](https://docs.unity3d.com/ScriptReference/Unity.Jobs.JobHandle.html). You can use a JobHandle in your code as a dependency for other jobs. If a job depends on the results of another job, you can pass the first job’s JobHandle as a parameter to the second job’s Schedule method, like so:

 **Combining dependencies**

If a job has many dependencies, you can use the method [JobHandle.CombineDependencies](https://docs.unity3d.com/ScriptReference/Unity.Jobs.JobHandle.CombineDependencies.html) to merge them. CombineDependencies allows you to pass them onto the Schedule method.

 **Waiting for jobs in the main thread**

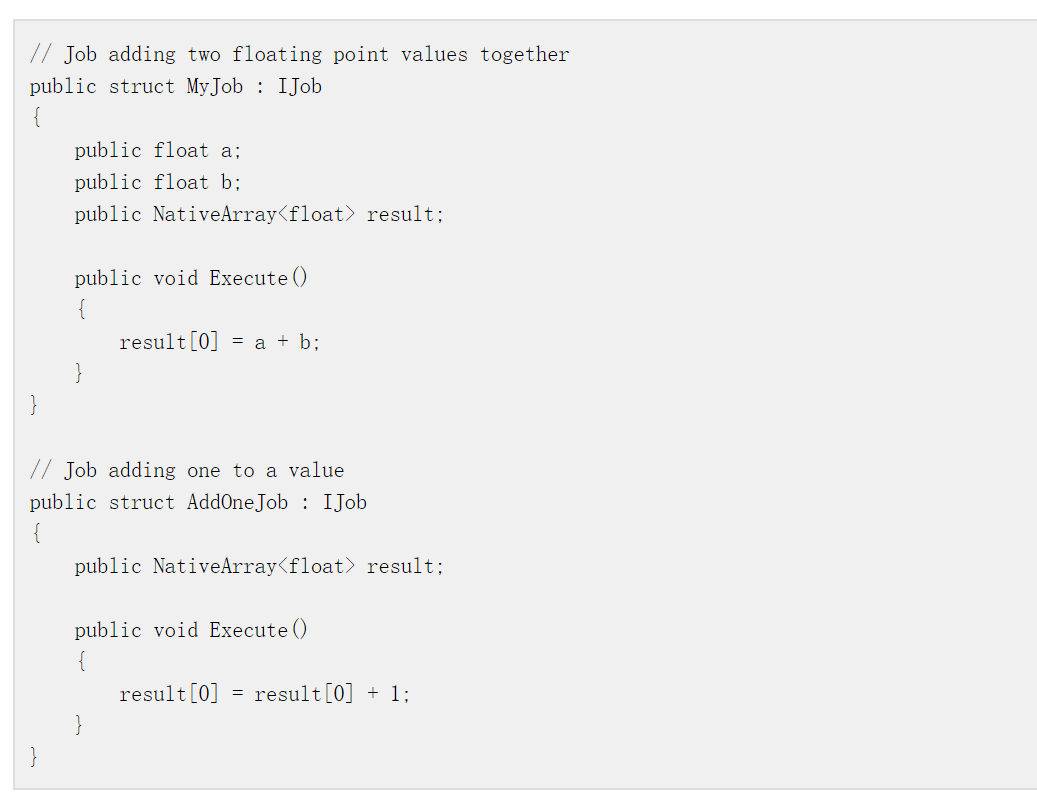
Use JobHandle to force your code to wait in the main thread for your job to finish executing. To do this, call the method [Complete](https://docs.unity3d.com/ScriptReference/Unity.Jobs.JobHandle.Complete.html) on the JobHandle. At this point, you know the main thread can safely access the [NativeContainer](https://docs.unity3d.com/ScriptReference/Unity.Collections.LowLevel.Unsafe.NativeContainerAttribute.html) that the job was using.

**Note**: Jobs do not start executing when you schedule them. If you are waiting for the job in the main thread, and you need access to the NativeContainer data that the job is using, you can call the method JobHandle.Complete. This method flushes the jobs from the memory cache and starts the process of execution. Calling Complete on a JobHandle returns ownership of that job’s NativeContainer types to the main thread. You need to call Complete on a JobHandle to safely access those NativeContainer types from the main thread again. It is also possible to return ownership to the main thread by calling Complete on a JobHandle that is from a job dependency. For example, you can call Complete on jobA, or you can call Complete on jobB which depends on jobA. Both result in the NativeContainer types that are used by jobA being safe to access on the main thread after the call to Complete.

Otherwise, if you don’t need access to the data, you need to explicity flush the batch. To do this, call the static method [JobHandle.ScheduleBatchedJobs](https://docs.unity3d.com/ScriptReference/Unity.Jobs.JobHandle.ScheduleBatchedJobs.html). Note that calling this method can negatively impact performance.

**An example of multiple jobs and dependencies**

**Job code**:

 **Main thread code**:

