**Async Overview**

Not so long ago, apps got faster simply by buying a newer PC or server and then that trend stopped. In fact, it reversed. Mobile phones appeared with 1ghz single core ARM chips and server workloads transitioned to VMs. Users still want responsive UI and business owners want servers that scale with their business. The transition to mobile and cloud and an internet-connected population of >3B users has resulted in a new set of software patterns.

* Client applications are expected to be always-on, always-connected and constantly responsive to user interaction (for example, touch) with high app store ratings!
* Services are expected to handle spikes in traffic by gracefully scaling up and down.

Async programming is a key technique that makes it straightforward to handle blocking I/O and concurrent operations on multiple cores. .NET provides the capability for apps and services to be responsive and elastic with easy-to-use, language-level asynchronous programming models in C#, VB, and F#.

**Why Write Async Code?**

Modern apps make extensive use of file and networking I/O. I/O APIs traditionally block by default, resulting in poor user experiences and hardware utilization unless you want to learn and use challenging patterns. Task-based async APIs and the language-level asynchronous programming model invert this model, making async execution the default with few new concepts to learn.

Async code has the following characteristics:

* Handles more server requests by yielding threads to handle more requests while waiting for I/O requests to return.
* Enables UIs to be more responsive by yielding threads to UI interaction while waiting for I/O requests and by transitioning long-running work to other CPU cores.
* Many of the newer .NET APIs are asynchronous.
* It's easy to write async code in .NET!