Advanced C# Concepts Cheat Sheet

Span<T> and Memory<T>

Span<T>: stack-allocated, lightweight struct for slicing arrays/memory. No heap allocation.
Memory<T>: similar to Span<T>, but can be used with async methods and lives on the heap.
Use for performance-critical scenarios like parsing or buffers.

Reflection

Provides metadata info at runtime (types, properties, methods). Use typeof(), GetType(), or System.Reflection namespace.
Can dynamically invoke methods, read attributes, etc. Slower than static code - use wisely.

Generics

Allow type-safe data structures and methods. Use constraints (where T : class, new()).
Generic interfaces and methods improve reusability and performance by avoiding boxing/unboxing.

Expression Trees

System.Linq.Expressions allows building code in tree form. Used in LINQ providers like Entity Framework to translate queries.
Can dynamically compile and execute code at runtime.

Roslyn (C# Compiler Platform)

Roslyn exposes C# compiler APIs. Enables code analysis, refactoring tools, custom analyzers.
Can parse syntax trees, walk semantic models, and emit diagnostics or transformations.

Dependency Injection (DI)

Design pattern for decoupling dependencies. Use .NET built-in Microsoft.Extensions.DependencyInjection.
Register services in IServiceCollection and inject via constructors.

Records and Init-only Properties

Records are immutable reference types. Great for value-like models.
Init-only properties (init keyword) allow setting props during object initialization but not after.

Nullable Reference Types (NRTs)

Enable with '#nullable enable'. Compiler gives warnings for potential null dereference.
Use string? for nullable references, and annotations like [NotNull], [MaybeNull].