C# Core Concepts Cheat Sheet

Garbage Collector

Managed by CLR. Automatically reclaims memory. Works in generations (0, 1, 2). Use GC.Collect() sparingly. Implement IDisposable to clean up unmanaged resources.

Delegate

Type-safe function pointer. Enables passing methods as parameters. Syntax: delegate void MyDelegate(string msg);

Event

Built on delegates. Used for publisher/subscriber model. Syntax: public event MyDelegate MyEvent;

EventHandler

Standard delegate for events. Signature: void(object sender, EventArgs e). Use EventHandler or EventHandler<T>.

Action & Func

Action: void return type, Func: returns a value. Func<int, string> maps int to string. Lambda expressions often used.

LINQ

Language Integrated Query. Enables querying collections. Supports method and query syntax. Works with IEnumerable and IQueryable.

IEnumerable vs IQueryable

IEnumerable: in-memory iteration, good for collections. IQueryable: builds expression trees, suitable for LINQ to SQL/EF.

List<T> vs IList<T>

List<T>: concrete type with full implementation. IList<T>: interface, more flexible for abstraction and testing.

IDisposable

Interface to free unmanaged resources. Implement Dispose(). Use 'using' statement or IAsyncDisposable for async cleanup.

Nullable Types

Value types like int can be made nullable using '?'. Example: int? age = null; Check HasValue or use ?? operator.

Async/Await

Simplifies async programming. Use Task<T> or ValueTask<T>. Await non-blocking I/O. Avoid blocking calls inside async methods.