.NET Cheat Sheet

C# · LINQ · Async/Await · Generics · NuGet

worksheetdownload.com

{ }

SETUP & CLI

```
dotnet new console -n MyApp
dotnet new webapi -n MyApi
dotnet add package Newtonsoft.Json
dotnet build  /  dotnet run  /  dotnet
test
dotnet publish -c Release -o ./publish
dotnet tool install -g dotnet-ef
```

BASIC SYNTAX

```
var name = "Alice";  // type inferred
string? maybeNull = null;  // nullable
reference type
int[] numbers = { 1, 2, 3 };
(string First, int Age) tuple = ("Bob",
30);
Console.WriteLine($"Hello {name}");
using var stream =
File.OpenRead("f.txt");  // auto-dispose
```

DATA TYPES

```
int long float double decimal bool char
string
byte short uint ulong
DateTime DateTimeOffset TimeSpan
Guid  (new Guid() or Guid.NewGuid())
object? anything = null;
dynamic d = "hello"; d = 42; // runtime
type
record Person(string Name, int Age); //
immutable by default
```

CLASSES & RECORDS

```
public class User {
public required string Name { get; init;
}
public int Age { get; set; }
}
// Record (value-equality, immutable):
public record Product(string Name,
decimal Price);
var p2 = p1 with { Price = 19.99m }; //
non-destructive update
```

COLLECTIONS

```
List<T>  Dictionary<K,V>  HashSet<T>
Queue<T>  Stack<T>  LinkedList<T>
IEnumerable<T>  IReadOnlyList<T>
var list = new List<int> { 1, 2, 3 };
var dict = new Dictionary<string,int> {{
"a",1 }};
Span<T>  Memory<T>   zero-copy slices
```

LINQ

```
var evens = numbers.Where(n => n % 2 ==
0);
var names = users.Select(u => u.Name);
var sorted = users.OrderBy(u =>
u.Age).ThenBy(u => u.Name);
var first = users.First(u => u.IsAdmin);
var groups = users.GroupBy(u => u.Role);
var flat = lists.SelectMany(l =>
l.Items);
users.Any() / .All() / .Count() / .Sum()
/ .Average()
```

ASYNC / AWAIT

```
public async Task<User> GetUserAsync(int
id)
{
var user = await _repo.FindAsync(id);
return user ?? throw new
NotFoundException();
}
var result = await Task.WhenAll(t1, t2,
t3);
await using var res = new
AsyncResource();
var cts = new
CancellationTokenSource(timeout:
TimeSpan.FromSeconds(5));
```

GENERICS

```
public T Find<T>(IEnumerable<T> items,
Func<T, bool> predicate)
=> items.FirstOrDefault(predicate);
public class Repository<T> where T :
class, IEntity
{ ... }
where T : struct  // value type
constraint
where T : new()   // must have
parameterless ctor
```

INTERFACES & INHERITANCE

```
public interface IRepository<T> {
Task<T?> FindAsync(int id);
Task SaveAsync(T entity);
}
public class UserRepo : BaseRepo,
IRepository<User> { ... }
sealed  // prevent further inheritance
abstract  // cannot be instantiated
```

NULLABLE REFERENCE TYPES

```
<Nullable>enable</Nullable>  // in
.csproj
string? nullableName = null;
string nonNull = nullableName ?? throw
new Exception();
if (name is not null) {
Console.Write(name.Length); }
name?.Length  // null-conditional
user!.Name    // null-forgiving (I know
it's not null)
```

COMMON GOTCHAS

async void = unobserved exceptions
always use async Task
LINQ is lazy evaluate with
.ToList()/.ToArray() when needed
ConfigureAwait(false) in library
code to avoid deadlocks
String comparison:
StringComparer.OrdinalIgnoreCase
DateTime.Now is local;
DateTime.UtcNow is UTC prefer UTC