1. OOP
2. Collections, Generic, Generic Collections
3. Events and delegates
4. File IO / Serialization
5. Reflection and Custom ~~Attributes~~ Attributes.
↳ a tool to read type-metadata dynamically and to invoke functionality of any assemblies at runtime.
↳ Using this feature we developed applns like VStudio intellicense, custom ORM
- F/W :- codefirst Approach → class Employee
↳ table [Emp]

.tt → template files.

↳ custom template

Emp tab tab

↳ public class Emp<T>

{

p. ~~Id~~

pub. Id

;

;

}

$$\begin{tikzcd}
\text{F/W :-} & 1.0 \arrow[r] & 2.0 \arrow[r] & 3.0 \arrow[r] \arrow[r, bend left] & 3.5 \arrow[r] \arrow[r, bend left] & 4.0 \\
\text{CLR :-} & 1.0 \arrow[r, dash] & 2.0 \arrow[r, dash] & 2.0 \arrow[r, dash] & 2.0 \arrow[r, dash] & 4.0
\end{tikzcd}$$

$\underbrace{2.0 \quad - \quad 2.0}_{\text{LINQ}}$

C# :- Year 2002

1.0 : 2002 ⇒ class✓, Properties✓, Events✓ and Delegates✓, struct✓

2.0 : 2005 ⇒ Partial Classes, Generic, Iterator, Nullable Types, Anonymous methods, Predicate delegate.

3.0 : 2008 ⇒ Implicit Type, Auto-Properties, Object Initializer, Anonymous Types, Lambda Expression, Extension methods → LINQ (Language Integrated Query]

4.0 : 2010 ⇒ Parallel Prog
- Task Parallel Library
- Parallel-for, Parallel-foreach loop
- Parallel-LINQ,

✓ Dynamic Type.

✓ Named and optional parameter

5.0 : 2012 $\Rightarrow$ Async and Await

6.0 : 2015 $\Rightarrow$ $" " : String Interpulation.

7.0 : 2017 $\Rightarrow$ out variable, Tuples

```
class Emp
{ Prop. Id, Nm, Add, DeptID
        int  str  str   int
}
List<Emp> emps
= new List<Emp>();

emps.Add( 1 | H | NY | 1 )

emps.Add( 2 | J | NJ | 2 )

emps.Add( new Emp()
    .Id = 5
    .Nm = "Rahul"
    .Add = "Pune",
    .DeptId = null !
```

ORM

- Hibernet (Java)
- Entity framework (.Net) [EF]

values

db.Table[Emp]

Id	Nm	Add	DP Id
1	Hugh	NY	1
2	Jack	NJ	2
3	Tom	An	4
4	Tim	Lendn	3
5	Rahul	Pune	

null

int = null

Nullable Types

```
Class Emp
{
    int ID;
  Nullable<int> Dept1d;
      int ? Dept2d;

}
```

:- Nullable types allows you to hold null values into value type containers / variables.

Auto - properties :-

private member will be generated by csharp compiler and it decides its name based on its naming algorithms.

Syntax :-

```
AccessModifier  Datatype  PropertyName
                   { set;   get; }
```

int [] arr = {1, 2, 3, 4, 5};

arr. Max() $\Rightarrow$ o/p: 5

~~Enumerable obj = new Enumerable();~~

obj. max (int[] arr)

Enumerable . Max (int [] arr)

this

arr. Max (this int[])

```
class    MyClass
{
        bool  CheckForValidEmail (string email)


}

                    obj= new MyClass()
MyClass obj= new MyClass()
      obj. CheckEmail (string email)

   string email =" a@ b.com";
         email. Check ( string.      )
```

public class Myclass : String

{

bool CheckEmail(this string email)

}

~~obj~~. Check.

email. Chrm

str = "mugdha@b.com";

str. Check