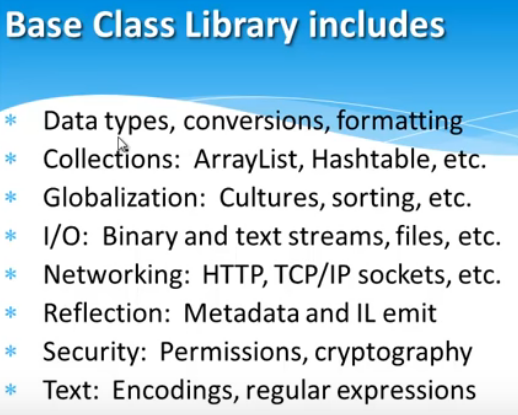
**What is .Net and why we used .Net**

* .Net is framework. Framework is like this is how it should be, it should be structural
* Basically its Microsoft tool which we used to develop software application
* Application types – console, Web, Windows, Mobile
* You can create services – web services, WCF, API’s
* It includes Libraries – predefined function means, if you want to print and display
* Previously mobile application & web application & Desktop application is used to develop in different languages. Say desktop application developed by c, C++ but using these languages, we cannot develop web application. That’s why .net comes into picture
* It provides set of programming languages like 30+ languages because different peoples coming from different background like c C++, java, Cobol – c#, vb.net, f#, j#
* Features - object oriented for security and reusability
* Platform independent for runs in windows, Android, IPhones
* Language independent for you can run c# code in vb.net application no need to install anything
* .net is free because java is free – you have to original license version of windows OS. Then you can download and run the .net framework which is supported by c#, vb.net all applications
* Versions 1.0 by 2002, 1.1 by 2003, 2.0 by 2005, 3.0 by 2006, 3.5 by 2007, 4.0 by 2010, 4.5 by 2012
* Why versions – introduced additional features in upcoming versions. From 1.0 to 1.1 nothing new features just fixed bugs. From 1.1 to 2.0 is generics, collections, architectural wise, new libraries, WCF, WPF. From 2.0 to 3.0 some additional features nothing much new. From 3.0 to 3.5 added LINQ, Ajax, and Entity Framework. From 3.5 to 4.0 is MVC

**Common Language Runtime**

* Its heart of .net
* Operating system understand 0 and 1. They cannot understand programing languages
* C# compiler convert programming language into IL( set of instructions which is run in any platform supported by .net). this is called assemblies and this format will be EXE or DLL
* IL cannot understand the OS so Converting from MSIL(Intermediate language) to OS language is used CLR with help of JIT compiler
* JIT compiler converts IL code into native code( 0,1)
* If you are running applications is Linux OS, CLR converts into their native code
* Native code not stored permanently. it will throws away once the application is closed
* That’s main advantage – platform independent, garbage collection
* Garbage collection – cleaning unwanted memories like employee object will allocate the memory at end of application we should clear that memory otherwise it will becomes memory space issues and sometimes application will not run

**Base Class Library**

* it is collection of object oriented classes and interface that provide services for many of the complex programing task that routinely used during the application development
* Ex: ADO.net class. It is part of class library. We use ado.net for connectivity from application to Database. When we need to create, update, delete operation no need to write programming for insert statements instead of just call create object from ado.net class which is already in libraries like that we can use it in update and delete as well
* We can use ado.net class library in many applications like Hospital, Hotel, Banking, Employee management application
* It’s like readymade either we can use it or we can extend it
* Language independent – in case if you are moving from c# to vb.net, just learn about syntax class library remains the same. Means what is syntax for forloop in vb.net that is it. So easily we can migrate from one programming language into another language
* 

**Framework class library**

* Predefined classes are library
* Set of predefined classes related to .net is called framework class library
* Predefined classes – already created by Microsoft
* More than 2000 predefined classes in FCL
* System.IO – file handling, System.data – connecting DB, System.text – text conversion, System.Web – classes for web development

**Assemblies**

* It is single unit of deployment
* Ex: if you copy exe and run on any platform it will work. That’s why it is single unit of deployment
* Compiler converts into two assemblies 1. DLL (dynamic) 2. EXE(static)
* When we compile class library DLL generate and when we compile Application EXE is generated
* DLL purpose – it is supportive file. Therefore, we can reuse much other application. Purpose is Reusability. It don’t have main method so We cannot execute
* EXE - you can see the output of application. It has main method so We can execute the exe
* Types – private and public
* Private – we can use for only single application. Location is folder itself bin/debug inside this folder you can find. When we use this assembly in multiple application, the copy of dll will be there. Same dll will be available in all application
* Public – use for multiple application. Location is C:\Windows\assembly.

Here dll will be placed in GAC(global assembly cache) and referred in all application, no copy of all dll in all application like private assembly.

**Referring Assemblies**

* Simply you can add reference to your project to get additional classes

**Common Type System**

* Deals with datatypes
* It specifies the datatypes rule regulation
* Ex: c# int where as vb.net interger finally CLR understand int32
* C# compiler is CSC (csharpcompiler)

**Common specification language**

* In .net we have multiple languages like c#, vb.net, f# … each end every language follows syntax like declaring variables like that. One language cannot understand other language syntax
* Ex: In c# each rows termination is ; but vb.net should not end with ;
* It specifies syntax rules regulation

**Interoperability between net languages**

* Code, which is written in any .NET Language, can be consumed in other .NET Language. This is called as Language Interoperability
* The CIL code which we get after compiling C# Language can be used by VB Language and the CIL Code which we get after compilation of VB code can be used by C#
* Language Interoperability is possible because even though Datatypes names are Different in Different .Net Languages their Sizes are same for all .Net Languages but after Compiling, they will generate same CIL Code which can be used by any other .Net Languages

**Garbage Collector and forcing the garbage collector**

* Garbage Collection makes automatic memory management possible in .NET Framework
* When a class object is created at runtime, certain memory space is allocated to it in the heap memory. However, after all the actions related to the object are completed in the program, the memory space allocated to it is a waste, as it cannot be used. In this case, garbage collection is very useful as it automatically releases the memory space after it is no longer required
* **Force** - If the ***GC.Collect*** method is called, then garbage collection occurs. However, this method is only called under unusual situations as normally garbage collector runs automatically

**Debugging**

* Debugging is the process of finding errors during application execution
* It does not mean syntax errors, with which the application cannot be compiled, but on logic errors.

**Database Connectivity & Types of Database connectivity**

* Accessing Data from a database is one of the important aspects of any programming language. It is an absolute necessity for any programming language to have the ability to work with databases. C# is no different.
* It can work with different types of databases. It can work with the most common databases such as Oracle and Microsoft[SQL](https://www.guru99.com/sql.html)Server.
* It also can work with new forms of databases such as[MongoDB](https://www.guru99.com/mongodb-tutorials.html)and MySQL.
* ADO.net
* System.Data.SqlClient - This assembly (namespace) of .NET Framework contains all of the classes required to connect to a SQL Server database and read, write, and update. The namespace provides classes to create a database connection, adapters, and sql commands that provides functionality to execute SQL queries
* Connection to a database requires a connection string. This string has the information about the server you're going to connect to, the database you will require and the credentials that you can use to connect
* **Data source**- Identify the server. Could be local machine, machine domain name, or IP Address

**Initial Catalog** - Data base name

**Integrated Security**- Set to SSIP to make connection with user's window login

**User ID** - Name of user configured in SQL Server

**Password** - Password matching SQL Server User ID

* **Types** – SQL, Oracle, Mango, MySQL