

بسم الله الرحمن الرحيم

الفهرس

Lesson 1 : About this Course & Installing C#

C# مميزات

- ❖ لغة C# تعتبر من أقوى لغات البرمجة وهي مطلوبة في سوق العمل
- ❖ تستطيع من خلالها إنشاء تطبيقات مختلفة مثل
 - Web development ○
 - Desktop development ○
 - Mobile development ○
 - وغيرها Embedded system ○
- ❖ Operating system يعني تستطيع تشغيل التطبيق على أي Portable language
- ❖ الهدف من تعلم تطبيقات Desktop قوي (frontend / backend developer) أن تكون



لأن .NET Framework هو إطار يوفر لك بيئة عمل لكل لغات البرمجة تحت مسمى .NET Framework ضمن إطار .NET Framework : تستطيع استخدام أكثر من لغة لإنشاء برنامج managed code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _14___C____Level_1
{
    internal class Program
    {
        static void Main(string[] args)
        {
            // cout in C++ على الشاشة مثل Syntax
            // Console.WriteLine("hi this is my first C# app :- ");
            // هذا الأمر لإيقاف الشاشة السوداء Console و عدم افالها
            // system("pause>0"); : C++ مثل الأمر في
            Console.ReadLine();
        }
    }
}
```

الفرق ما بين Library vs Framework vs Platform

١. Library هي عبارة عن مجموعة من Functions التي لها علاقة مع بعضها البعض في مكان واحد "مكتبة" سواء كان كل

a. Function .b.

أو ضمن Class

ليتم استدعائهما في أي مشروع من غير إعادة كتابة Function مثل مكتبة Math

٢. Framework هو أوسع من Library وهو مكون من مجموعة من المكتبات والأدوات

والقوانين التي تحكم Developer في كيفية كتابته للكود - بيئة عمل تجعل Developers

يعملون على نسق واحد في كتابة الكود - "بيئة لكتابة البرنامج"

٣. Platform أوسع من Framework وهي بيئة تشغيل البرنامج مثل Windows , MacOS

أو hardware هي البيئة التي يتم تشغيل البرنامج فيها سواء كانت environment .a

What is Library?

A library is a collection of pre-written code that developers can use to add functionality to their applications.

A library typically provides a set of functions or classes that a developer can call from their code to perform specific tasks, without having to write all the code from scratch.

Example: Math Library.



What is Framework?

A framework is a set of libraries and tools that provide structure and support for building and running applications.

A framework defines a set of rules, protocols, and conventions that developers must follow when writing their code. This helps ensure that all parts of an application work well together and follow a consistent design pattern.

What is Platform?

A platform refers to the hardware or software environment in which a piece of software runs.

For example, a computer running the Windows operating system could be considered a platform for running software written for Windows

Platform = Programming Language + Libraries

Lesson 3 : What is .NET ? .NET Core vs .NET Framework ? What can we do with .NET ?

معلومات عامة عن .NET

- ❖ Programming Language + Libraries = **Platform** هي مكونة من
- ❖ أمثلة على لغات البرمجة التي تدعم .NET : C# , Visual Basic , F# ...
- كل هذه اللغات التي في .NET تستخدم نفس المكتبات
- كل هذه اللغات لها أداء واحد performance لأنهم يستخدموا نفس المكتبات
- الاختلاف في هذه اللغات **Syntax فقط** أما الأداء والسرعة متماثلين فيهما - على حسب الخوارزمية O
- ❖ .NET هو منصة كبيرة Platform فيها مكتبات وأدوات تمكّنك من إنشاء التطبيقات
- فيه أكثر من **Net Platform** نوع :
- **.Net Core**
 - وجد حديثاً هو Open-source
 - وميزته أنه يدعم البرنامج على أي Operating system مثل Windows , MacOS , Linux
 - أمثلة : Console , ASP .NET Core
 - اسم المكتبة الأساسية : .NET Core BCL (base class library)
- **.Net Framework**
 - هو أول نوع وجد من .NET.
 - يدعم البرنامج على نظام Windows فقط سواء Desktop apps
 - تطبيقات سطح المكتب
 - خدمات Services
 - مواقع Websites
- أمثلة : WPF , Windows Forms , Console , ASP.NET
- اسم المكتبة الأساسية : .NET Framework BCL (base class library)
- Flutter : يدعم تطبيقات الجوال Xamarin / Mono
 - أمثلة على التطبيقات: ios , Mac OS , Android
 - اسم المكتبة الأساسية : Mono BCL (base class library)
- ❖ يوجد في Platform مكتبة اسمها .NET Standard
- وهذه المكتبة يوجد فيها APIs¹ وهو عبارة عن مكتبات تستخدم لجميع Net Platform
- ❖ ما سيتم دراسته - بإذن الله - تستطيع تطبيقه على أي نوع من أنواع Platform

ما هي التطبيقات التي يمكن ببناؤها ب.NET؟ لا يهم بأي لغة برمجة

❖ تستطيع بناء أي تطبيق باستخدام .NET.
Desktop , web , cloud , Mobile , Gaming , IOT , Ai ❖

What is .NET?



Platform = Programming Language + Libraries

.Net is a Platform ☺

Languages in .Net Platform are:

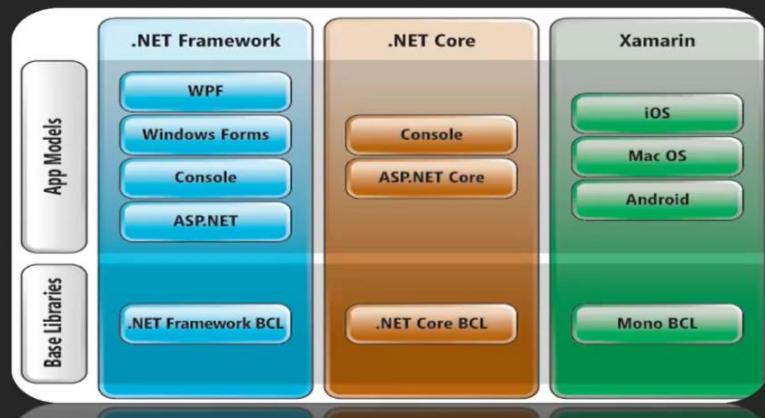
- C#, Visual Basic, and F#...etc.

Platforms:

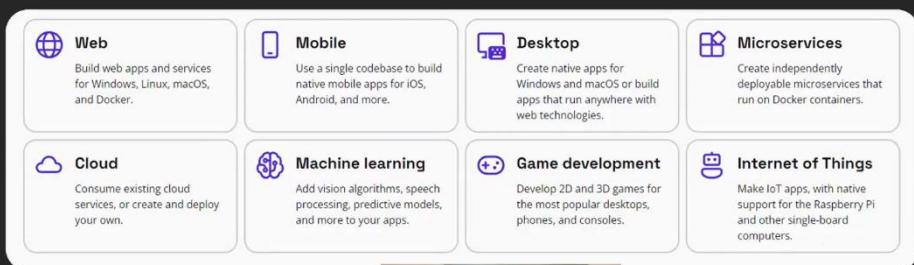
- .Net Core: (runs anywhere) Windows, Linux, and macOS
- .Net Framework: Websites, services, and desktop apps ..etc on windows
- Xamarin /Mono : a .NET for Mobile (Cross Platform runs on android or ios)

All use a standard set of libraries ☺

.NET Platform?



What can we build using .NET?



Who Built .NET?

Microsoft + several thousands of companies ☺

Over 100,000 developers and 3700 Companies Contribute to the .NET Platform ☺

.NET Platform

- .NET is a platform, created by Microsoft, for building and running many different types of applications including web, desktop, mobile, gaming, IoT, AI, and cloud.
- It is a comprehensive development platform that provides tools and technologies for building all types of applications, including web, desktop, mobile, gaming, IoT, AI, and cloud.
- The .NET platform includes multiple components such as the .NET runtime, the Base Class Library (BCL), and the .NET Standard Library.

.NET Standard

- .NET Standard is a formal specification of the APIs that are common across .NET implementations. This allows the same code and libraries to run on different implementations.

.NET Framework

- ".NET Framework" is one of the components of the .NET platform.
- It is a specific implementation of the .NET platform that provides a runtime environment for executing .NET applications and provides a set of libraries for developing and running applications on Windows.
- The .NET Framework was first released in 2002 and has since been updated multiple times to include new features and improvements.

In short, .NET is a platform, while .NET Framework is one of its implementations.

.Net vs .Net Core



- ".NET" and ".NET Core" are both components of the .NET platform, but they differ in their design goals and use cases.
- ".NET Core" is a cross-platform, open-source, and modular implementation of the .NET platform. It was created to provide a high-performance, scalable, and flexible runtime environment for building modern, cloud-based applications that can run on multiple operating systems, including Windows, macOS, and Linux.
- .NET Core provides a subset of the features and libraries available in the full .NET platform, but it is designed to be faster, lighter-weight, and more efficient.
- In short, .NET is a comprehensive platform for building all types of applications, while .NET Core is a cross-platform, open-source, and modular implementation of the .NET platform focused on modern, cloud-based applications.

لما ندرس .NET Core . وليس .NET Framework ؟

- ❖ أحدث إصدار من .NET هو .NET Core. في عام 2016م ويدعم جميع أنواع Operating system وهو مفتوح المصدر
- ❖ بينما .NET Framework يدعم جميع التطبيقات لكن على Windows فقط
- ❖ **من أكبر أخطاء تعلم الطلاب الجدد للبرمجة هو تعلم : أحدث التكنولوجيا**
 - يؤدي ذلك الى عدم إيجاد عمل .NET Framework
 - لأن معظم الشركات بنو أنظمتهم على .NET Framework
 - ويطلب سنوات لنقل نظامهم الى أحدث إصدار .NET Core. تقريرا من 5 الى 10 سنوات
 - ويطلب مال وجهد وقد يرفض بعض العملاء التغيير
 - قد تكتب المشاريع الجديدة ب .NET Core. أما القديمة يصعب إعادةتها
 - لأن الشركات تحتاج موظفين يعرفون أنظمتهم
 - بعد تعلم .NET Framework بشكل جيد ، تعلم .NET Core. ولن يأخذ منك وقت كثير لأنها نفس المكتبات
 - التكنولوجيا الجديدة تأخذ سنوات للتحول إليها بدل القديمة

كيفية عمل Compilation في .NET ؟

هو **Platform .NET** ويوجد بها لغات برمجة عديدة وكل هذه اللغات تشارك في طريقة عمل **Compilation**

❖ يوجد نوعين ترجمة **Code** الى² **Machine²**

- Interpreter
- Compiler

أما .NET فتدعم كلاهما

١. طريقة عمله مثل **JIT** Interpreter هذا يعمل بشكل افتراضي

٢. طريقة عمله مثل **AOT** Compiler

يوجد في **Platform .NET** لغات عديدة لهم ميزة مشتركة وهي : **Managed Code** معناه أنها لغات تتم إدارتها من قبل مدير وهو **CLR** اختصار Common Language Runtime :

❖ كل اللغات التي في **Platform .NET** لها نفس السرعة لأنها تستخدم نفس المكتبات

❖ وكل لغة لها **Compiler** خاص بها – لا يتم التحويل مباشر الى Machine Language – وإنما يتم التحويل الى

توحيد كل اللغات الى **Common Intermediate Language** اختصار **IL / CIL** ○
لغة وسطية

❖ ثم يتم تحويلها الى المدير **CLR** اختصار Common Language Runtime : بداخله **JIT**

اختصار : **Just In Time Compiler** ○

Interpreter ○ يشبه عمل **JIT**

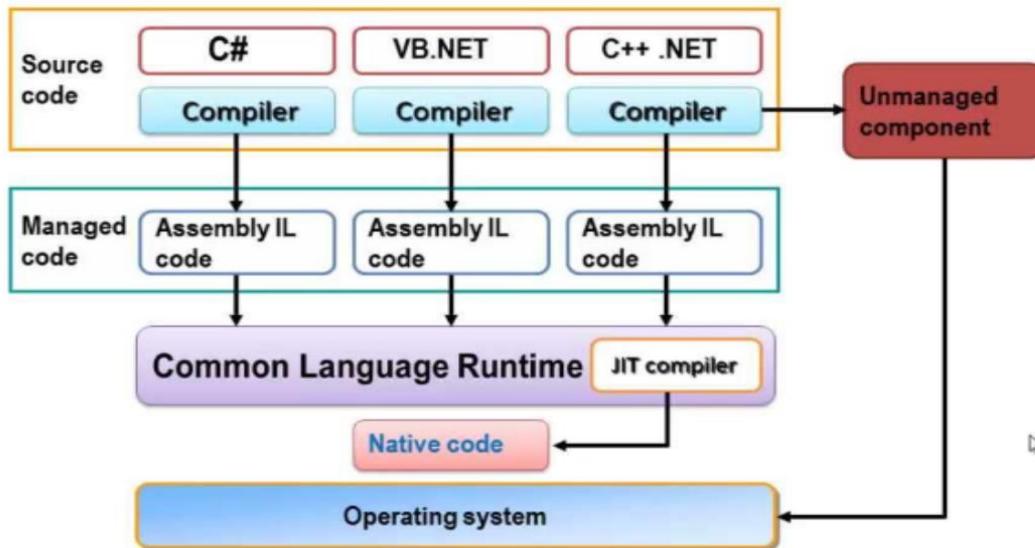
- ومسؤوليته تحويل الكود التي تحتاجه الان بعمل **Compilation** الى **Flexibility** في **Run Time** في **Machine Language**
- يستهلك ذاكرة أكثر

❖ ويوجد آخر **AOT Mode** اختصار **AOT** ○ يشبه طريقة عمل **Compiler**

○ ويتم إلغاء عمل **JIT**

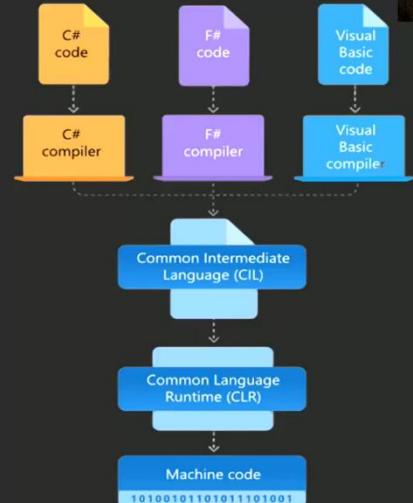
² تم شرح الفروق في الكورس 1 الدرس 12

The CLR Execution Model



Compilation

- .NET applications are written in the C#, F#, or Visual Basic programming language. Code is compiled into a language-agnostic Common Intermediate Language (CIL). Compiled code is stored in assemblies—files with a .dll or .exe file extension.
- When an app runs, the CLR takes the assembly and uses a just-in-time compiler (JIT) to turn it into machine code that can execute on the specific architecture of the computer it is running on.



Compilation

In .NET, the compilation process occurs in two stages:

- just-in-time (JIT) compilation.
- ahead-of-time (AOT) compilation.

JIT: JUST In Time Compilation.

- Just-in-Time (JIT) Compilation: During the execution of a .NET application, the CLR performs JIT compilation of the application's Intermediate Language (IL) code into machine code. The JIT compiler translates the IL code into native machine code for the target platform on which the application is running. This allows the application to take advantage of the full performance of the target platform's hardware. The JIT compiler only compiles the methods that are actually executed, so applications start up quickly, and unused code is never compiled.

AOT: Ahead of Time Compilation.

- AOT stands for Ahead-of-Time, which is a method of compiling .NET code into machine-readable code that can be executed directly by the computer's processor, without requiring an interpreter or just-in-time (JIT) compilation. This approach can lead to faster start times and better performance compared to JIT-compiled code.
- AOT-compiled .NET code is often used in scenarios where performance and startup time are critical, such as in mobile or embedded devices, or in cloud environments where instances are frequently restarted. The AOT-compiled code is optimized for the specific architecture and can take advantage of hardware features such as instruction set extensions and hardware acceleration.
- In .NET, AOT-compilation is usually performed using the Native Image Generator (Ngen.exe) tool, which creates a native image from an existing .NET assembly. The native image can then be deployed and executed on the target machine, without requiring the .NET runtime to be installed.

.NET Platform. كان من الأفضل تسميتها .NET Framework ❖
هو قواعد وقوانين لتنظيم العمل بين Developers ، وفيه مكتبات .NET Framework ❖
لا يستطيع تشغيل البرنامج .NET Framework ❖
وتسميته ب .NET Framework. يشمل .NET Platform. ويستطيع تشغيل البرنامج .NET Framework ❖
من أقوى الموجدة في البرمجة Platform .NET Framework ❖

- .NET. يتكون من شيئين رئيسيين - مشترك في جميع .NET Framework Architecture

١. اختصاراً : Common Language Runtime : هو الذي يشغل البرنامج
٢. فيه مجموعة هائلة من Classes وهي Class Library in .NET
٣. تستطيع استخدامها بأي لغة تدعم .NET Framework .a. لغات البرمجة Languages

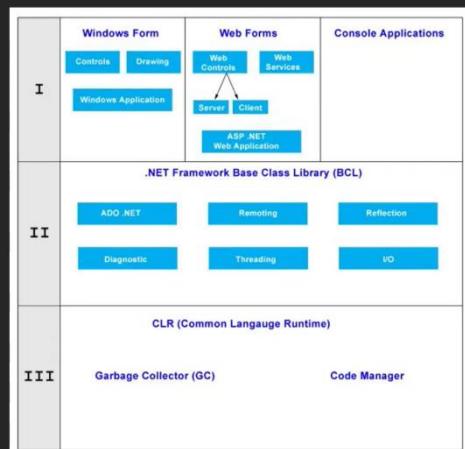
Architecture of .NET Framework

The two major components of .NET Framework are the Common Language Runtime and the .NET Framework Class Library.

- The Common Language Runtime (CLR) is the execution engine that handles running applications. It provides services like thread management, garbage collection, type-safety, exception handling, and more.
- The Class Library provides a set of APIs and types for common functionality. It provides types for strings, dates, numbers, etc. The Class Library includes APIs for reading and writing files, connecting to databases, drawing, and more.

Another important component is : Languages.

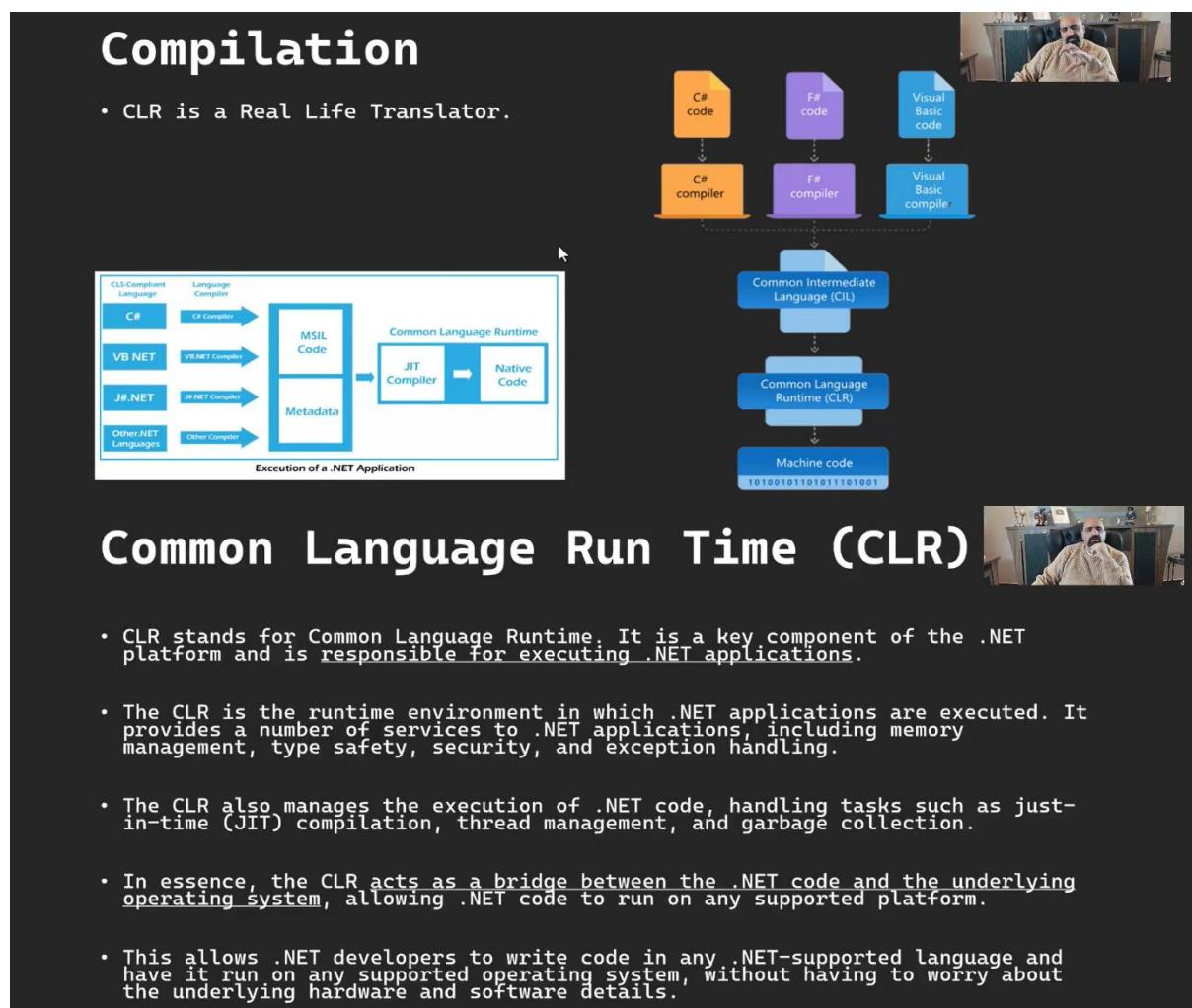
Architecture of .NET Framework



Lesson 7 : What is CLR ?

ما هو CLR ؟

- ❖ هو أحد Commons الرئيسية التي في .NET Framework
- ❖ CLR هو المسؤول عن تنفيذ (executing) في تطبيقات .NET
- ❖ وكل لغة لها Compiler خاص بها – لا يتم التحويل مباشر إلى Machine Language – وإنما يتم التحويل إلى
- اختصار Common Intermediate Language : توحيد كل اللغات إلى لغة وسطية
- لها اسم آخر وهو MSIL اختصار
- Common Language Runtime : اختصار
- ❖ ثم يتم تحويلها إلى المدير CLR عن طريق Machine Language عن طريق الذي يحولها من IL / CIL إلى
- Interpreter Just In Time Compiler : يشبه عمل JIT Compilation
- ومسؤوليته تحويل الكود التي تحتاجه الآن بعمل Compilation إلى Flexibility Run Time Machine Language



المكونات الرئيسية في CLR

- ❖ Common Type System : CTS
- ❖ Common Language Specification : CLS
- ❖ Garbage Collector : GC
- ❖ Just In Time Compiler : JIT
- ❖ Metadata
- ❖ Assemblies

- ❖ من المكونات الرئيسية ل CLR هو CTS اختصاراً
- ❖ Common Type System
- ❖ يدعم عدة لغات وكل لغة يوجد فيها Data type مختلف في syntax.NET Framework
- ❖ عند تشغيل Compilation البرنامج يتم تحويل الكود الى لغة وسطية IL / CIL / MSIL
- ❖ Common Intermediate Language
- 1. مثال int في C# و integer في VB يتم دمجها في Data Type جديدة تحت مسمى Type Save Language تسمى Int32
- ❖ لذلك تستطيع إنشاء برنامج بأكثر من لغة لأنه يتم توحيد Data Type في أخرى جديدة

Common Type System (CTS):



CTS provides guidelines for declaring, using, and managing data types at runtime.

It offers cross-language communication.

For example, VB.NET has an integer data type, and C# has an int data type for managing integers. After compilation, Int32 is used by both data types.

So, CTS provides the data types using managed code. A common type system helps in writing language-independent code.

Lesson 10 : 2- Common Language Specification (CLS)

- ❖ من المكونات الرئيسية ل CLR هو **CLS** اختصار ل Common Language Specification
- ❖ **CLS** : المواصفات العامة التي لابد من توفرها في اللغة لتصبح من ضمن .NET Framework
- ❖ لن تصبح اللغة من ضمن **NET Framework** حتى تتبع القواعد والقوانين التي في **CLS** وهذه هي مهمة **CLS**
- ❖ **NET Framework** يدعم عدة لغات وكل لغة يوجد فيها **Data type** مختلفة في **syntax** ولكن كل اللغات تتبع القواعد والقوانين التي في **CLS**
- ❖ **NET Framework** يجعل هناك توافق **Interoperability** بين اللغات بمعنى آخر - تكامل - فهم اللغات لبعضها البعض
 ١. مثال انشاء برنامج على VB واستدعاء Function من C#
 ٢. عند تشغيل البرنامج على Debugging mood ينتقل من VB الى C# بكل سلاسة وتوافق وتكامل بينهما وهذه هي **CLS**

Common Language Specification (CLS)

Common Language Specification (CLS) contains a set of rules to be followed by all .NET-supported languages.

The common rules make it easy to implement language integration and help in cross-language inheritance and debugging.

Each language supported by .NET Framework has its own syntax rules.

But CLS ensures interoperability among applications developed using .NET languages.

Lesson 11 : 3- Garbage Collector (GC)

- ❖ من المكونات الرئيسية ل CLR هو **GC** اختصار ل Garbage Collector
- ❖ **وظيفة GC الرئيسية** هو تنظيف الذاكرة غير المستخدمة بشكل تلقائي لإعادة استخدامها
- ❖ **GC** موجود في اللغات الحديثة واللغات التي تدعم .NET
- ❖ لا يوجد في لغة C++ : GC خصوصاً عند استخدام `(delete new)` يقابل `new`
- ❖ **GC** يساعدك على إدارة الذاكرة بينما في C++ يكون بشكل يدوى

Garbage Collection:



Garbage Collector is a component of CLR that works as an automatic memory manager.

It helps manage memory by automatically allocating memory according to the requirement.

It allocates heap memory to objects.

When objects are not in use, it reclaims the memory allocated to them for future use.

It also ensures the safety of objects by not allowing one object to use the content of another object.

Lesson 12 : 4- Just In Time Compilation (JIT)

- ❖ من المكونات الرئيسية ل CLR هو **JIT** اختصار ل : Just In Time Compilation
- ❖ عند تشغيل **Compilation** البرنامج يتم تحويل الكود الى لغة وسطية **IL / CIL / MSIL**
- ❖ كل لغة لها **Compiler** خاص بها – لا يتم التحويل مباشر الى Machine Language – وإنما يتم التحويل الى **IL / CIL / MSIL** أصبحت لدينا لغة جديدة
- ❖ ثم يتم تحويلها الى **CLR** الذي يحولها من **IL / CIL / MSIL** عن طريق **JIT** الذي يحولها الى Native Code
- ❖ **أنواع JIT يوجد ثلاثة أنواع :**

١ . **AOT Ahead Of Time** الذي يسمى **Pre JIT Compiler**

- مهمته عمل **Compiler** لكل الكود وتخزينه في ملف **Ngen.exe**
- لا يوجد **Compilation** أثناء تشغيل البرنامج
- يستهلك ذاكرة كبيرة مثل **C++**
- لكنه أسرع لأنّه لا يوجد **Compilation** أثناء **Run Time**

٢ . **Econo JIT** : يعمل **Compiler** للّكود الذي بحاجته مع كل استدعاء في كل مرة ولكن لا

يتم تخزينه – يستهلك ذاكرة قليلة ولكنه بطيء

٣ . **Normal JIT (Default)**

- مهمته عمل **Compilation** للّكود الذي بحاجته الآن وتخزينه **cache memory**
- مثال : شاشة تستدعي **Function1** يتم عمل **Function1** **Compilation** وتخزينه في الذاكرة وعند استدعاء **Function1** من شاشة أخرى لا يعمل له ، **Cache Memory** وإنما يتم جلبه من **Compilation** ولا يتم تخزين أي **Function** آخر لم يتم استدعاؤه

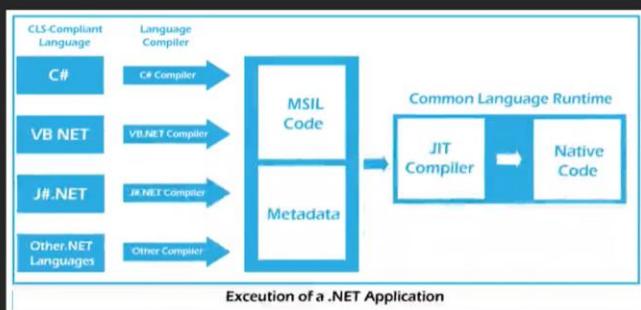
- يستهلك ذاكرة أقل بكثير من AOT Ahead Of Time
 - لكنه أبطأ لأنّه يوجد أثناء Compilation
 - ❖ من الأسرع بينهم لا يوجد جواب عام لأنّه يكون على حسب البرنامج
١. عادة يكون للبرامج الكبيرة بشكل عام **Normal JIT** يكون أسرع وقد يكون **Econo JIT**
 ٢. وللبرامج الصغيرة يكون **AOT** Ahead Of Time

Just in Time (JIT) Compiler:

JIT Compiler is an important component of CLR.



It converts the MSIL code into native code (i.e., machine-specific code).



Just in Time (JIT) Compiler:



The .NET program is compiled either explicitly or implicitly.

There are three types of JIT compilers -Pre, Econo, and Normal.

1. Pre JIT Compiler (AOT Ahead Of Time): compiles entire MSIL code into native code before execution.
2. Econo JIT: Compiler compiles only those parts of MSIL code required during execution and removes those parts that are not required anymore.
3. Normal JIT (Default): Compiler also compiles only those parts of MSIL code required during execution but places them in cache for future use. It does not require recompilations of already used parts as they have been placed in cache memory.

Just in Time (JIT) Compiler:



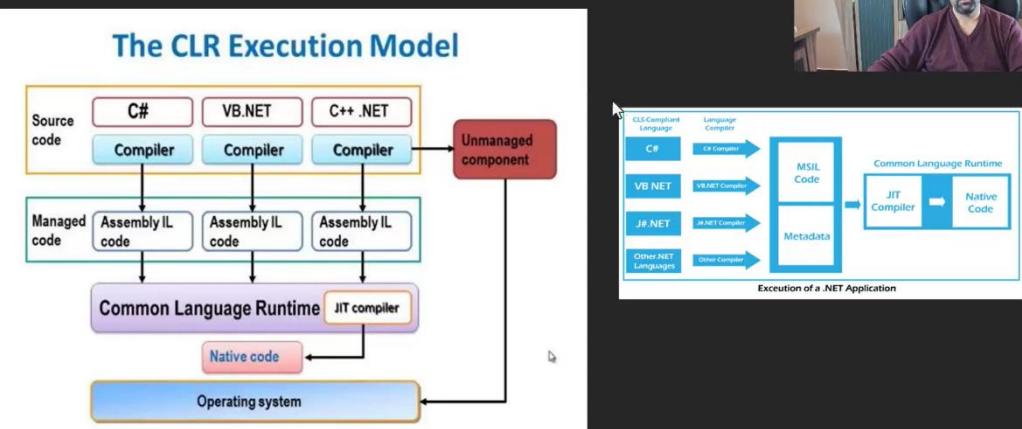
The .NET program is compiled either explicitly or implicitly.

There are three types:

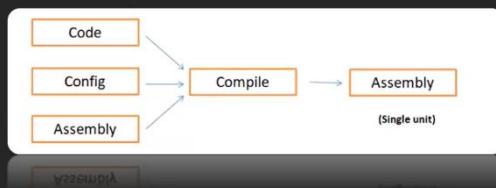
1. Pre JIT Compiler (AOT Ahead Of Time): (Ngen.exe) , everything compiles before start, No compilation at runtime.
2. Econo JIT: Compiled and not cached (each time it will compile again).
3. Normal JIT (Default): only used code is Compiled + Cached.

- ❖ من المكونات الرئيسية ل CLR هو **Assembly**
 - ❖ من المكونات الرئيسية ل CLR هو **Metadata**
 - ❖ كل لغة لها **Compiler** خاص بها – لا يتم التحويل مباشر الى Machine Language – وإنما يتم التحويل الى **IL / CIL / MSIL** أصبحت لدينا لغة جديدة وتسما **Assembly IL Code**
 - ❖ وتسما أيضا **single Unit** (ونتيجة الملف في C++ اسمه **exe.file** للتقرير)
 - ❖ لا يوجد بداخله كود وإنما نتيجة **Source Code**
 - ❖ (**Assembly (single Unit**) – الملف exe – هو الذي يتم تنزيله أو تثبيته عند الزيون
 - ❖ يتم تجميعهم في **Unit** وتسما **Source Code**
 - ❖ عند عمل مشروع يوجد ملفات معينة يتم تثبيتها عند الزيون وتسما **Deployment**
 - ❖ نتائج **Assembly** واحد منها
- ١ . وهي ملف exe تستطيع تشغيله **Process Assembly**
 - ٢ . وهي ملف DLL اختصار ل **Library Assembly**
- وهي عبارة عن برنامج كامل لكن لا تستطيع تشغيله بمفرده وإنما Reference الى Desktop / Web / Mobile
 - وهي Class Library – سيدرس في 2 C# Level
 - مكوناته من أي (مكتبات ، لغة)
 - من الكاتب author / الإصدار version / الاسم / تاريخه ...
 - وهذه المعلومات يحتاجها NET. للرجوع إليها إذا احتاجها
 - وهذه المعلومات ضرورية ل JIT للتعامل مع **Assembly (single Unit**)
- ❖ ثم يتم تحويلها الى CLR الذي يحولها من **IL / CIL / MSIL** الى Machine Language عن طريق JIT الذي يحولها الى Native Code على نظام التشغيل

Assemblies:



Assemblies:

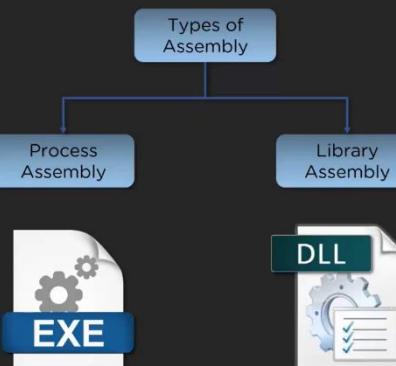


An assembly is a fundamental unit of physical code grouping.

- It is basically a compiled code that can be executed by the CLR.
- Assembly is a single unit of deployment.
- Pre-Compiled Chunk of code that can be executed by CLR.
- It is also considered a basic deployment unit.

Assemblies:

Assembly can refer to either a dynamic link library (DLL) or an executable file (.exe), depending on its intended use.



Metadata (Manifest):

Metadata is a Data that provides information about other data.

Metadata is information about the assemblies, modules, and types that constitute .NET programs.

Metadata (Manifest):

Metadata Contains:

- The assembly's name
- Version number
- Digital signature that uniquely identifies an assembly's creator.
- All files which build up the assembly
- Information regarding all of the referenced assemblies.
- Information of all the exported classes, methods, properties, and other items.

- ❖ .NET هو المدير العام لجميع لغات التي **CLR**.
- ❖ من أهم وظيفة لـ **CLR** هي تحويل من **Native Code** إلى **IL / CIL / MSIL**.
- ❖ مسؤول عن الأمان
- ❖ يوجد أكثر من Platform في .NET. وكلها تستخدم نفس المكتبات ...

.NET CLR Functions

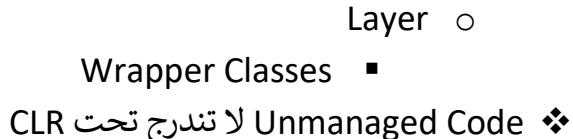
- It converts the program into native code.
- Handles Exceptions.
- Provides type-safety.
- Memory management.
- Provides security.
- Improved performance.
- Language independent.
- Platform independent.
- Garbage collection.
- Provides language features such as inheritance, interfaces, and overloading for object-oriented programs.
- Manager for all .NET supported languages.



❖ كل لغات .NET framework تعتبر **Managed Code** لأنه يكون تحت " كل هذه اللغات تفهم بعضها البعض "

❖ وبقي اللغات التي لا تدعم .NET framework تسمى **Unmanaged Code** مثل java و C++ العادية (يوجد C++ تدعم .NET)

❖ يوجد في CLR أدوات للتعامل مع **Unmanaged Code**



Managed Code

- The code that runs with CLR is called managed code, whereas the code outside the CLR is called unmanaged code.
- The CLR also provides an Interoperability layer, which allows both the managed and unmanaged codes to interoperate.
- Any language that is written in the .NET framework is managed code. Managed code uses CLR, which looks after your applications by managing memory, handling security, allowing cross-language debugging, etc.

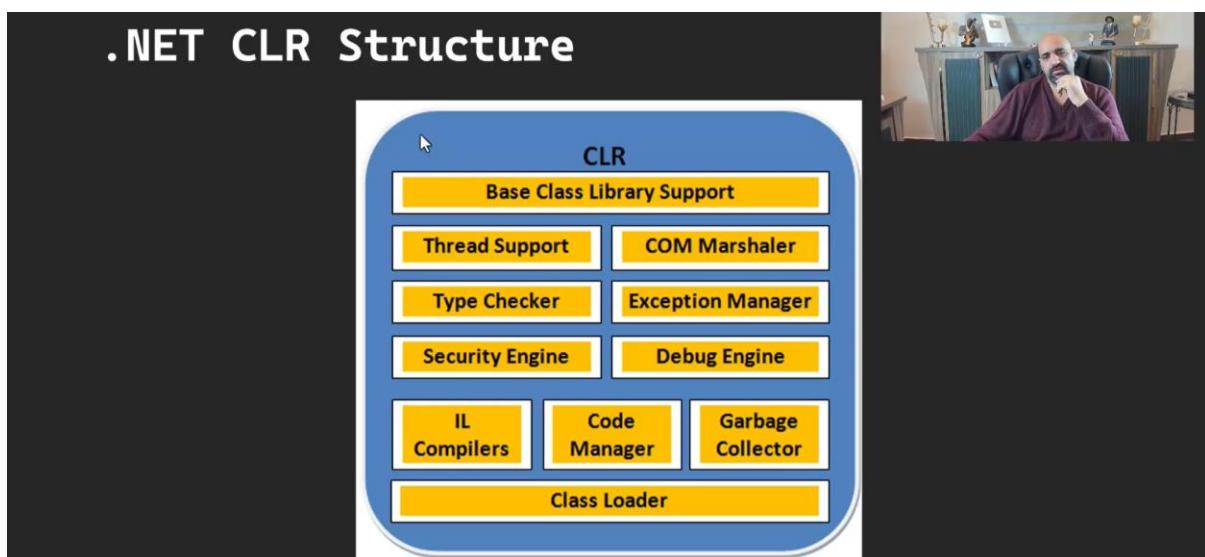
Unmanaged Code



- The code developed outside the .NET framework is known as unmanaged code.
- Applications that do not run under the control of the CLR are said to be unmanaged. Certain languages such as C++ can be used to write such applications, such as low-level access functions of the operating system.
- Background compatibility with VB, ASP, and COM are examples of unmanaged code. This code is executed with the help of wrapper classes.

COM: Component Object Model: It's a client server communication using method in the server without the need to compile them into your application. (old technology 1993 by Microsoft).

- ❖ دعم للمكتبات التي في .NET Base Class Library Support
- ❖ دمج أكثر من Function3 (Support / Programming)
- مثال : Function1 يجمع أرقام Function2 فتستطيع استدعاؤهم معاً
- ❖ وظيفته التأكد من هذه اللغة أو Data type متبعة لقوانين .NET Type Checker
- ❖ بالنسبة ل Types Framework
- ❖ .NET framework مسؤول عن الأخطاء بين لغات البرمجة في Exception Manager



Base Class Library

It is a class library that supports classes for the .NET application.

Thread Support

It manages the parallel execution of the multi-threaded application.

COM Marshaler

It provides communication between the COM objects and the application.

Security Engine

It enforces security restrictions.

Debug Engine

It allows you to debug different kinds of applications.

Type Checker

It checks the types used in the application and verifies that they match the standards provided by the CLR.

Code Manager

It manages code at execution runtime.

Garbage Collector

It releases the unused memory and allocates it to a new application.

Exception Handler

It handles the exception at runtime to avoid application failure.

Lesson 17 : .NET Framework Class Library (FCL)

- ❖ هو عبارة عن مجموعة من Classes وعددها بالآلاف – كل Class له عمل معين كود جاهز – وتسمى **Base Class Library**
- ❖ وتدعم أنواع **.NET**. سواء **Desktop , web , Mobile**
- ❖ وهذه المكتبات تدعم كل شيء

.NET Framework Class Library

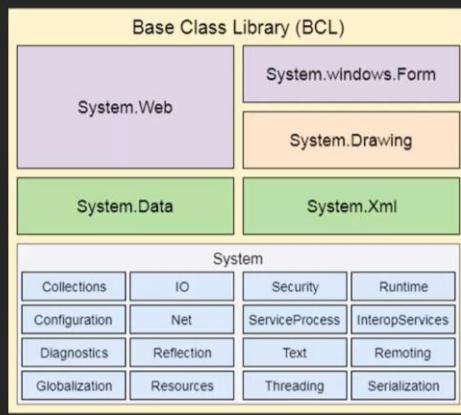
.NET Framework Class Library is the collection of classes, namespaces, interfaces and value types that are used for .NET applications.

It contains thousands of classes that supports the following functions.

- Base and user-defined data types
- Support for exceptions handling
- input/output and stream operations
- Communications with the underlying system
- Access to data
- Ability to create Windows-based GUI applications
- Ability to create web-client and server applications
- Support for creating web services



.NET Framework Class Library



لابد من فهم الدروس السابقة بدون تعمق في هذا المستوى – مهم جدا –
والتي تم دراسته هو الأساس المشترك في جميع أنواع **.NET**. بشكل عام

لغة C# مشابه جداً لـ Java و C++ للغة Syntax

سيتم التركيز على الفروقات والاختلافات ولن يتم شرح المبادئ (for loop – if , else ...)

ما هي C# وبعض مميزاتها ؟ (pronounced as C sharp)

- ❖ هي OOP فقط وليس FP يعني أنك لا تستطيع كتابة Function إلا داخل OOP
- ❖ وهي لغة حديثة عام 2002م قوية جداً / تستطيع فهم الكود بسرعة
- ❖ وهي مبنية على لغة C , C++
- ❖ تستطيع من خلالها إنشاء تطبيقات قوية وبسرعة (أسرع من C++)
- ❖ تتضمن كل متغير في مكانه – رقم في int & النص في string
- ❖ وهي من ضمن اللغات التي تدعم .NET

لماذا سميت ب C# ؟ الذي اخترعها سماها على نوطة موسيقية

ما الذي يمكن إنشاءه ب C# ؟

- كل التطبيقات التي تستطيع بناءها باستخدام .NET
- Desktop , web , cloud , Mobile , Gaming , IOT , Ai

What is C#?



- C# (pronounced as C sharp) is a general-purpose, object-oriented programming language.
- The first version was released in year 2002. The latest version, C# 11, was released in November 2022.
- It is one of the most popular languages used for developing desktop and web applications.
- Being a C based language, C# is closer to C++ and C. Syntactically, it is similar to Java.
- Modern – C# is a modern and powerful language that allows developers to build robust applications quickly and easily. It is built based on the current trend

What is C#?

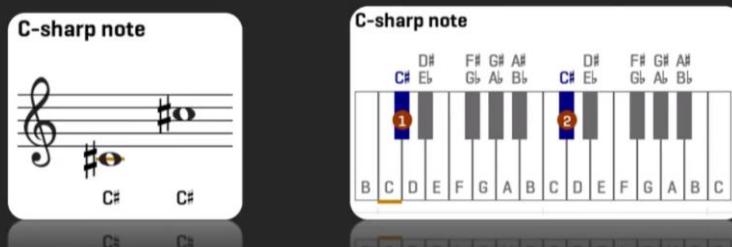


- **Simple** – The code written in C# is much simpler and easier to understand. It is syntactically very similar to Java.
- **Type Safe** – C# ensures that each variable of a particular type does not hold values of other types.
- **Object-oriented** – C# supports the object-oriented paradigm such as objects, classes, inheritance, polymorphism, etc.
- Being a high-level language, the basic constructs of C# is easy to understand.
- C# has a huge community. Hence a C# related question is more likely to be answered quickly

Why it's called C#?

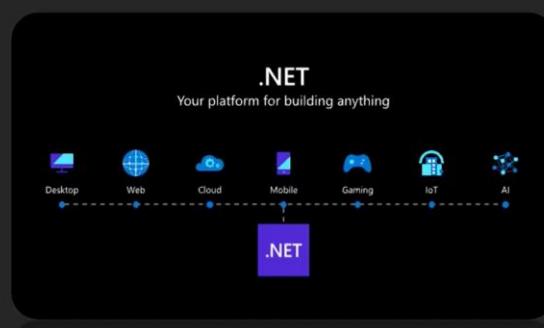


The name "C sharp" was inspired by the musical notation whereby a sharp symbol indicates that the written note should be made a semitone higher in pitch.



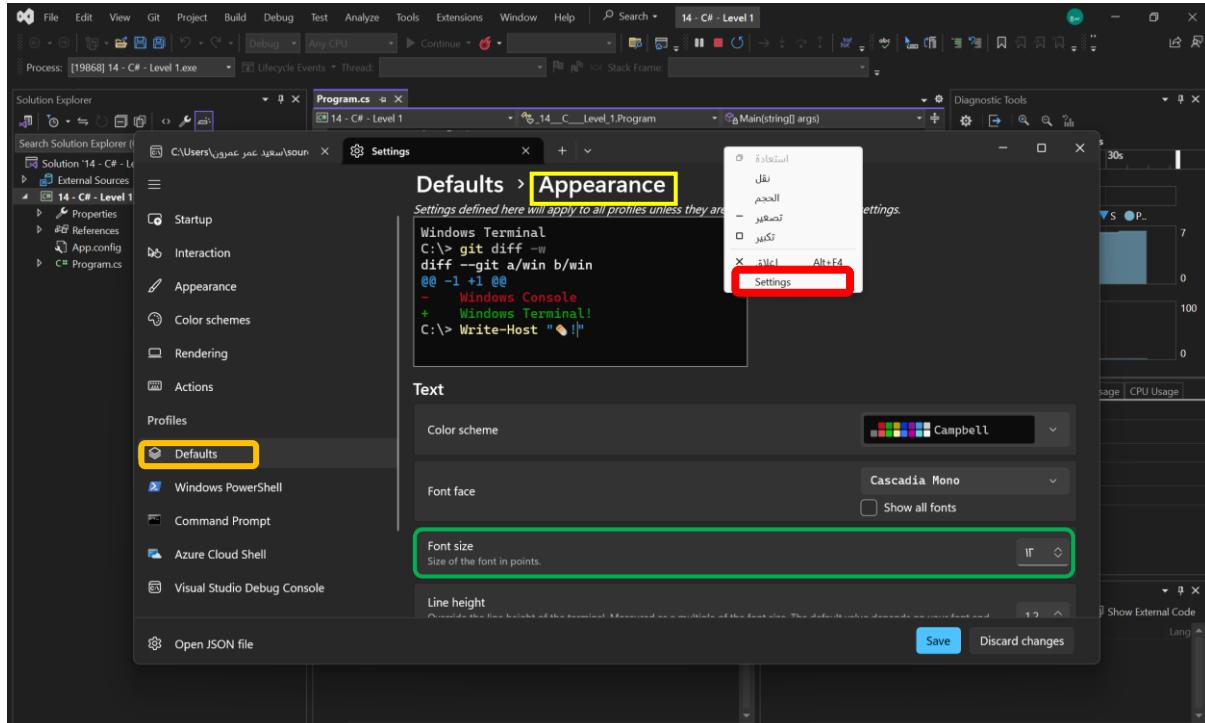
What can we do with C#?

Everything we can do with .NET ☺



Lesson 19 : Syntax

لتكبير الخط في شاشة Console



كل Function يكتب تحت Class لأن C# هي Fully OOP فقط - حتى main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

using System;

namespace Main
{
    internal class Program
    {
        // Object يتم استدعاؤه من غير static
        static void Main(string[] args)
        {
            // cout in C++ للطباعة على الشاشة مثل هذا هو Syntax
            //Console.WriteLine("Hello World!");

            // هذا الأمر لإيقاف الشاشة السوداء Console و عدم إغلاقها
            // مثل الأمر في C++ : system("pause>0");
            //Console.ReadKey();
        }
    }
}
```

Lesson 20 : WriteLine

في C++ مع النزول بسطر جديد في نهاية الجملة cout << " C++ \n"; مثل WriteLine

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

using System;

namespace Main
{
    internal class Program
    {
        // Object يتم استدعاؤه من غير static
        static void Main(string[] args)
        {
            // cout in C++ على الشاشة مثل Syntax
            Console.WriteLine("Hello World!");
            Console.WriteLine("My Name is Mohammed Abu-Hadhoud");
            Console.WriteLine("I am Learning C#");
            Console.WriteLine("It is an easy language to learn :-)");

            // 30 = ( 10 + 20 ) لابد من كتابتها داخل قوسين (
            // إذا لم تكتب داخل قوسين فنكون = 1020
            Console.WriteLine("The sum of 10 + 20 is " + (10 + 20));

            // هذا الأمر لإيقاف الشاشة السوداء و عدم افالها
            // مثل الأمر في C++ system("pause>0");
            Console.ReadKey();
        }
    }
}
```

Lesson 21 : Write

في C++ مع عدم النزول بسطر جديد في نهاية الجملة cout << " C++ "; مثل Write

```
Console.Write("I am Learning C#");
Console.Write("It is an easy language to learn :-)");
Console.Write("The sum of 10 + 20 is " + (10 + 20));
```

Lesson 22 : Formatted String

عمل **printf** في C++ مثله في ال String **Format**³
ال التعويض عن parameter بشكل تلقائي في مكان {0} , {1} ...
مثال ل C#

```
// تعويض عن { الرقم بعد الفاصلة يبدأ من 0 } ل كل رقم بعد الفاصلة له {0} Index
Console.WriteLine("{0} {1}", "Welcome to", "ProgrammingAvices");
Console.WriteLine("{1} {0}", "Welcome to", "ProgrammingAvices");
Console.WriteLine("Hi MyName is: {0} I live in {1}", "Mohammed", "Jordan");
```

مثال ل C++

```
char Name[] = "Mohammed Abu-Hadhoud";
char SchoolName[] = "Programming Advices";

printf("Dear %s, How are you ? \n\n", Name);
printf("Welcome to %s school \n\n", SchoolName);
```

Lesson 23 : Escape Characters (\)

C++ في تماثل عملها في C# **Escape Characters (\)**

ترجع خطوة الى الوراء - حذف الحرف الذي قبلها (\b \n \t \' \" \\ \a)

```
Console.WriteLine("Useful Escape Characters:\n");
//Newline
Console.WriteLine("Newline:");
Console.WriteLine("Welcome to \n ProgrammingAvices\n");
//Tab
Console.WriteLine("Tab:");
Console.WriteLine("Welcome to\tProgrammingAvices\n");
//Backspace
Console.WriteLine("Backspace:");
Console.WriteLine("Welcome to \bProgrammingAvices\n");
//Single quote
Console.WriteLine("Single Quote:");
Console.WriteLine("Welcome to \' ProgrammingAvices\n");
//Double quote
Console.WriteLine("Double Quote:");
Console.WriteLine("Welcome to \" ProgrammingAvices\n");
//Backslash
Console.WriteLine("Backslash:");
Console.WriteLine("Welcome to \\ ProgrammingAvices\n");
//Alert
Console.WriteLine("Alert:");
Console.WriteLine("\a");
```

³ مراجعة الكورس 6 الدرس 25

Lesson 24 : Single Line/Multiple Lines Comments (// , /**)

لكتابة التعليقات في C# مثل كتابة التعليقات في C++

// تعليق على سطر واحد

/* تعليق على عدة أسطر */

Lesson 25 : Variables

تعريف C# في Variables مثل تعريف C++ في Variables

أما double يضاف في آخر القيمة D

```
string MyName = "Mohammed Abu-Hadhoud";
Console.WriteLine(MyName);

int x = 10; int y = 20;

Console.WriteLine("x=" + x);
Console.WriteLine("y=" + y);

//this line will give wrong answer :-
Console.WriteLine("x+y=" + x + y);

//this line will give right answer :-
Console.WriteLine("x+y=" + (x + y));

//other common data types
double MyDouble = 25.89D;
char MyLetter = 'M';
bool MyBool = true;
```

Lesson 26 : Rules for Naming Variables in C#

كل قواعد تسمية Variables مثل C++
و C# تدعم رمز آخر وهو (_) فقط

Rules of Naming Variables

- The variable name can contain letters (uppercase and lowercase), underscore(_) and digits only.
- The variable name must start with either letter, underscore or @ symbol,
Examples:

Rules for naming variables in C#	
Variable Names	Remarks
name	Valid
subject101	Valid
_age	Valid (Best practice for naming private member variables)
@break	Valid (Used if name is a reserved keyword)
101subject	Invalid (Starts with digit)
your_name	Valid
your name	Invalid (Contains whitespace)



Rules of Naming Variables

- C# is case sensitive. It means age and Age refers to 2 different variables.
- A variable name must not be a C# keyword. For example, if, for, using can not be a variable name. keywords can found in the link below:

<https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/>

C# Keywords in the following link :

<https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords>

Lesson 27 : Implicitly Typed Variables

⁴ C++ في Automatic Variables يشابه C# Implicitly Typed Variables

- ❖ الأفضل في تعريف Typed Variables هي استخدام نوع المتغير المراد استخدامه مثل (int , string , float) يكون أسرع للبرنامج
- ❖ الأفضل عدم استخدام المتغير التلقائي var
- ❖ لابد من وضع قيمة مبدئية للمتغير التلقائي var

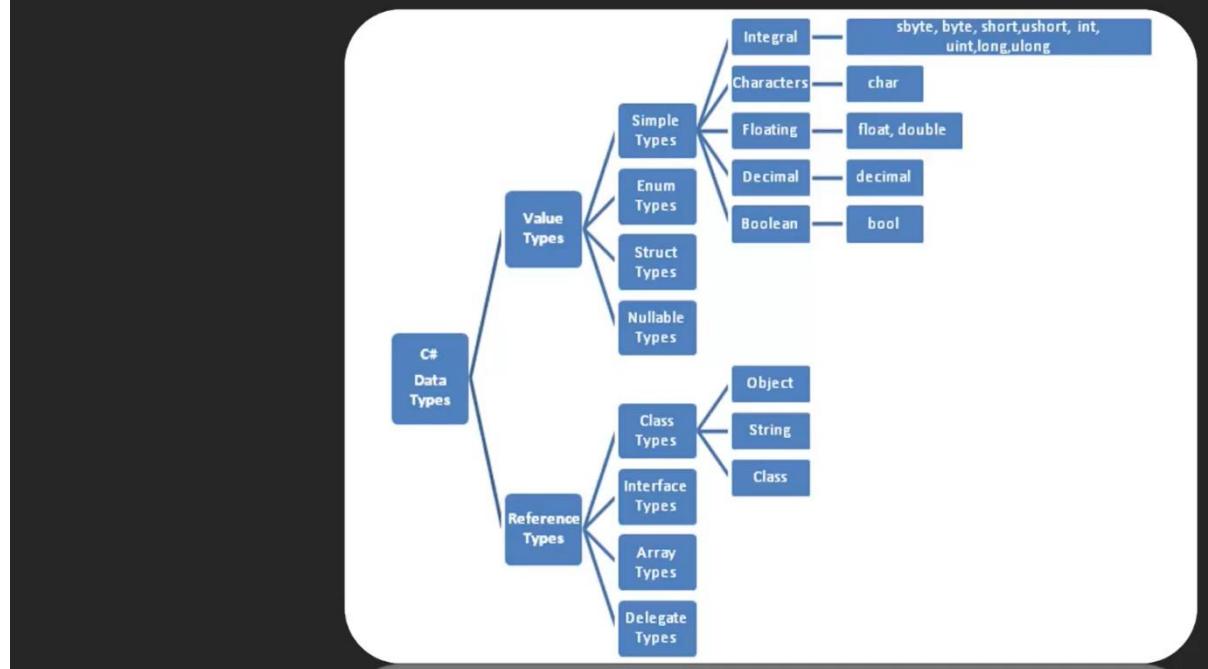
```
/* Implicitly typed variables
Alternatively in C#, we can declare a variable without knowing
its type using var keyword.
Such variables are called implicitly typed local variables.

Variables declared using var keyword must be initialized at the time
of declaration.
*/
var X = 10;
var Y = 10.5;
var Z = "Mohammed";

Console.WriteLine("x={0}, y={1}, z={2}", X, Y, Z);
```

Lesson 28 : Datatypes

Datatypes In C#



Lesson 29 : Predefined Datatypes

❖ في C# يتم تحويلها الى لغة جديدة لها **Datatypes** مشتركة بين جميع اللغات

التي في .NET. وتسمى هذه اللغة **Assembly IL Code**

❖ أصغر **Datatype** في C# هي **byte** وحجمها 255

- بينما أصغر **Datatype** في C++ هي **short** وحجمها 32767

❖ في C# يوجد **Suffix** وهو إضافة حرف خاص الى نهاية الرقم لتحديد نوع **Datatype** بشكل صريح

- **uint U = 5u / long L = 10L / ulong UL = 20UL**

- **float F = 25.10f / double d = 30.20d / decimal m = 500m**

Datatypes In C#

Type	Description	Size (bytes)	.NET type	Range
int	Whole numbers	4	System.Int32	-2,147,483,648 to 2,147,483,647
long	Whole numbers (bigger range)	8	System.Int64	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	Floating-point numbers	4	System.Single	+/-3.4 x 10^38
double	Double precision (more accurate) FP numbers	8	System.Double	+/-1.7 x 10^308
decimal	Monetary values	16	System.Decimal	28 significant figures
char	Single character	2	System.Char	N/A
bool	Boolean	1	System.Boolean	True or False
DateTime	Moments in time	8	System.DateTime	0:00 on 01/01/0001 to 23:59:59 on 12/31/9999
string	Sequence of characters	2 per character	System.String	N/A

Predefined Data Types:

Type	Description	Range	Suffix
byte	8-bit unsigned integer	0 to 255	
sbyte	8-bit signed integer	-128 to 127	
short	16-bit signed integer	-32,768 to 32,767	
ushort	16-bit unsigned integer	0 to 65,535	
int	32-bit signed integer	-2,147,483,648 to 2,147,483,647	
uint	32-bit unsigned integer	0 to 4,294,967,295	u
long	64-bit signed integer	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	l
ulong	64-bit unsigned integer	0 to 18,446,744,073,709,551,615	ul
float	32-bit Single-precision floating point type	-3.402823e38 to 3.402823e38	f
double	64-bit double-precision floating point type	-1.79769313486232e308 to 1.79769313486232e308	d
decimal	128-bit decimal type for financial and monetary calculations	(+ or -)1.0 x 10e-28 to 7.9 x 10e28	m
char	16-bit single Unicode character	Any valid character, e.g. a, *, \x0058 (hex), or \u0058 (Unicode)	
bool	8-bit logical true/false value	True or False	
object	Base type of all other types.		
string	A sequence of Unicode characters		
DateTime	Represents date and time	0:00:00am 1/1/01 to 11:59:59pm 12/31/9999	

الأرقام تنقسم الى قسمين

١. الأرقام الصحيحة Integer Types

- (0 to 255) **8 bit** = **byte** .a
- (-128 to 127) **8 bit** = **sbyte** .i
- (-32768 to 32767) **16 bit** = **short** .b
- (0 to 65535) **16 bit** = **ushort** .i
- (0 to 2 مiliyar to 2 مiliyar) **32 bit** = **int** .c
- (0 to 4 مiliyar) **32 bit** = **uint** .i
- 64 bit** = **long** .d
- 64 bit** = **ulong** .i

٢. الأرقام الكسرية floating Point Types

- 32 bit** = **float** .a
- 64 bit** = **double** .b
- 128 bit** = **decimal** .c

❖ لمعرفة حجم أكبر وأصغر قيمة يمكن تخزينها في Datatype

Assembly IL Code في اللغة الوسيطة ○ تكتب اسم Datatype

مثلاً : ... ■

.MinValue or .MaxValue ■

❖ تستطيع تخزين int في hexadecimal & Binary

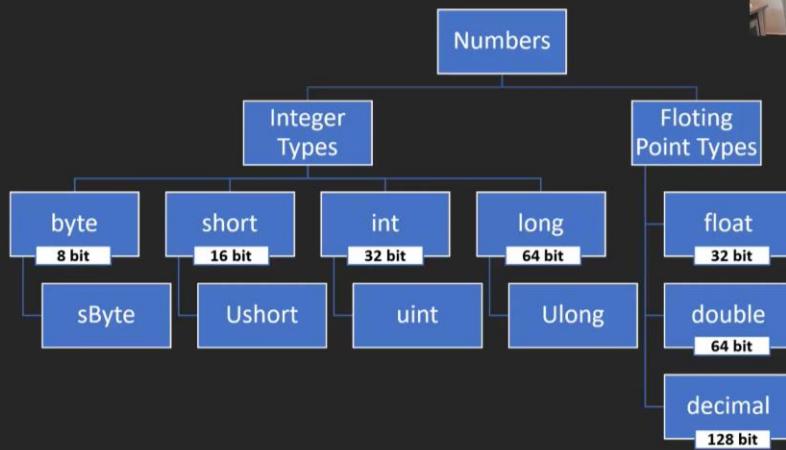
تكون بداية القيمة **0x** ○

مثال : int hex = 0x2F ; ■

تكون بداية القيمة **0b** ○

مثال : int binary = 0b_0010_1111 ; ■

Numbers Datatypes



```
//Byte
byte b1 = 255;
// byte b2 = -128;// compile-time error: Constant value '-128'
cannot be converted to a 'byte'
sbyte sb1 = -128;
sbyte sb2 = 127;
Console.WriteLine("\nByte:");
Console.WriteLine("Min={0} , Max={1}", Byte.MinValue, Byte.MaxValue);

Console.WriteLine("\nSByte:");
Console.WriteLine("Min={0} , Max={1}", SByte.MinValue,
SByte.MaxValue);

//Short
short s1 = -32768;
short s2 = 32767;
// short s3 = 35000;//Compile-time error: Constant value '35000'
cannot be converted to a 'short'

ushort us1 = 65535;
// ushort us2 = -32000; //Compile-time error: Constant value '-32000' cannot be converted to a 'ushort'

Console.WriteLine("\nShort:");
Console.WriteLine("Min={0} , Max={1}", Int16.MinValue,
Int16.MaxValue);

Console.WriteLine("\nUShort:");
Console.WriteLine("Min={0} , Max={1}", UInt16.MinValue,
UInt16.MaxValue);

//int
int i = -2147483648;
int j = 2147483647;
// int k = 4294967295; //Compile-time error: Cannot implicitly
convert type 'uint' to 'int'.
```

```

        uint ui1 = 4294967295;
        // uint ui2 = -1; //Compile-time error: Constant value '-1' cannot be
converted to a 'uint'

        Console.WriteLine("\nInt:");
        Console.WriteLine("Min={0} , Max={1}", Int32.MinValue,
Int32.MaxValue);

        Console.WriteLine("\nUInt:");
        Console.WriteLine("Min={0} , Max={1}", UInt32.MinValue,
UInt32.MaxValue);

        //Long
long l1 = -9223372036854775808;
long l2 = 9223372036854775807;

ulong ul1 = 18223372036854775808ul;
ulong ul2 = 18223372036854775808UL;

        Console.WriteLine("\nLong:");
        Console.WriteLine("Min={0} , Max={1}", Int64.MinValue,
Int64.MaxValue);

        Console.WriteLine("\nULong:");
        Console.WriteLine("Min={0} , Max={1}", UInt64.MinValue,
UInt64.MaxValue);

        //Float
float f1 = 123456.5F;
float f2 = 1.123456f;

        Console.WriteLine("\nFloat:");
        Console.WriteLine("Min={0} , Max={1}", float.MinValue,
float.MaxValue);

        //double
double d1 = 12345678912345.5d;
double d2 = 1.123456789123456d;

        Console.WriteLine("\nDouble:");
        Console.WriteLine("Min={0} , Max={1}", double.MinValue,
double.MaxValue);

        //Decimal
//The decimal type has more precision and a smaller range
//than both float and double,
//and so it is appropriate for financial and monetary calculations.
decimal d3 = 123456789123456789123456789.5m;
decimal d4 = 1.1234567891345679123456789123m;

        Console.WriteLine("\nDecimal:");
        Console.WriteLine("Min={0} , Max={1}", decimal.MinValue,
decimal.MaxValue);

        //Scientific Notation
//Use e or E to indicate the power of 10
//as exponent part of scientific notation with float, double or
decimal.

```

```

double d = 0.12e2;
Console.WriteLine(d); // 12;

float f = 123.45e-2f;
Console.WriteLine(f); // 1.2345

decimal m = 1.2e6m;
Console.WriteLine(m); // 1200000

//hex & Binary
int hex = 0x2F;
int binary = 0b_0010_1111;

Console.WriteLine(hex);
Console.WriteLine(binary);

```

Lesson : Default Values

لكل نوع من Datatype له قيمة افتراضية

- ❖ 0 = Numbers
- ❖ False = bool
- ❖ '0\' = char

– الإصدار الجديد – default = Name Datatype

```

//get default value using default(type)
int i = default(int); // 0
float f = default(float); // 0
decimal d = default(decimal); // 0
bool b = default(bool); // false
char c = default(char); // '\0'

// C# 7.1 onwards
//get default value using default
int i2 = default; // 0
float f2 = default; // 0
decimal d2 = default; // 0
bool b2 = default; // false
char c2 = default; // '\0'

```

Lesson : Enum

C++ فيEnum مثل C# فيEnum

```
using System;

namespace Main
{
    internal class Program
    {

        enum enWeekDays
        {
            Monday,      // 0
            Tuesday,     // 1
            Wednesday,   // 2
            Thursday,    // 3
            Friday,      // 4
            Saturday,    // 5
            Sunday       // 6
        }

        //if you set a value , it will auto number everything after it.
        enum enCategories
        {
            Electronics, // 0
            Food,        // 1
            Automotive = 6, // 6
            Arts,         // 7
            BeautyCare,   // 8
            Fashion       // 9
        }

        //Enum can have any numerical data type byte, sbyte, short, ushort, int,
        uint, long, or ulong
        //but not string

        enum enCategories2 : byte
        {
            Electronics = 1,
            Food = 5,
            Automotive = 6,
            Arts = 10,
            BeautyCare = 11,
            Fashion = 15
        }

        static void Main(string[] args)
        {

            enWeekDays WeekDays;
            WeekDays = enWeekDays.Friday;

            Console.ReadKey();
        }
    }
}
```

Lesson : Nullable Types

لا يمكن إنشاء قيم فارغة null لأنواع البيانات مثل ; int i = null خطأ
يمكن استخدام قيم فارغة لأنواع البيانات أخرى مثل ; Name = null
(Datatype ? Name = null ; وهي يوجد طريقة أخرى)

≡ Nullable Types

As you know, a value type cannot be assigned a null value. For example, `int i = null` will give you a compile time error.

C# 2.0 introduced nullable types that allow you to assign null to value type variables. You can declare nullable types using where T is a type. `Nullable<t>`

```
Nullable <int> i = null;
```

example, `Nullable<int>` can be assigned any value from -2147483648 to 2147483647, or a null value.

see the code.

```
// Nullable<int> can be assigned any value
// from -2147483648 to 2147483647, or a null value.

Nullable<int> i = null;

الطريقة الأخرى //  
int? r = null;
```

Lesson : Anonymous Type

يسمح بإنشاء object من غير Class – العادي ❖

❖ الأسماء التي بداخلها يتم تعريفها مباشرة من غير Datatype

❖ **Read Only** هو للقراءة فقط لا تستطيع أن تغير القيم ❖

❖ طريقة كتابته ; Id = 20 } ;

❖ تستطيع طباعة قيمة محددة فيه Student.Id

❖ تستطيع طباعة { كل ما بين الأقواس } باستدعاء نفسه Anonymous Type

❖ و تستطيع إنشاء داخلAnonymous Type أكثر من واحد

```
//you don't specify any type here , automatically will be specified
var student = new { Id = 20, FirstName = "Mohammed", LastName = "Abu-Hadhoud" };

Console.WriteLine("\nExample1:\n");
Console.WriteLine(student.Id); //output: 20
Console.WriteLine(student.FirstName); //output: Mohammed
Console.WriteLine(student.LastName); //output: Abu-Hadhoud

//You can print like this:
Console.WriteLine(student);

//anonymous types are read-only
//you cannot change the values of properties as they are read-only.

// student.Id = 2;//Error: cannot change value
// student.FirstName = "Ali";//Error: cannot change value

//An anonymous type's property can include another anonymous type.
var student2 = new
{
    Id = 20,
    FirstName = "Mohammed",
    LastName = "Abu-Hadhoud",
    Address = new { Id = 1, City = "Amman", Country = "Jordan" }
};

Console.WriteLine("\nExample2:\n");
Console.WriteLine(student2.Id);
Console.WriteLine(student2.FirstName);
Console.WriteLine(student2.LastName);

Console.WriteLine(student2.Address.Id);
Console.WriteLine(student2.Address.City);
Console.WriteLine(student2.Address.Country);
Console.WriteLine(student2.Address);
```

مُثُلٌ Syntax في C++ مع اختلاف بسيط Structure ♦

- يتم تعريف **Public Datatype** قبل **Public Class**
- تستطيع إنشاء متغير

- عادي : لا بد من وضع قيمة مبدئية - قبل الطباعة مثلا - وإن حدث **error**
- ليس شرطاً أن تضع قيمة مبدئية = **new**

لوضع **Default Constructor** وإنما هي **pointer** new •

قيمة مبدئية **Default** على كل **Datatypes** داخل **Structure**

• تخزن في **Stack** وليس **Heap** مثل **C++**

❖ يُستخدم لتخزين البيانات الصغيرة التي لا تتطلب الوراثة **Structure**

```
using System;
namespace Main
{
    internal class Program
    {
        struct stStudent
        {
            public string FirstName;
            public string LastName;
        }

        static void Main(string[] args)
        {

            //A struct object can be created with or without the new operator,
            //same as primitive type variables.

            stStudent Student;
            stStudent Student2 = new stStudent();

            Student.FirstName = "Mohammed";
            Student.LastName = "Abu-Hadhoud";

            Console.WriteLine(Student.FirstName);
            Console.WriteLine(Student.LastName);

            Student2.FirstName = "Ali";
            Student2.LastName = "Ahmed";

            Console.WriteLine(Student2.FirstName);
            Console.WriteLine(Student2.LastName);

            Console.ReadKey();
        }
    }
}
```

Lesson : Dynamic Type

يسمح بتخزين أي نوع من Datatype سواء كانت القيمة رقم أو حروف أو غيرها **Dynamic Type**

يتم تعريفها : dynamic Name = value

```
dynamic MyDynamicVar = 100;
Console.WriteLine("Value: {0}, Type: {1}", MyDynamicVar,
MyDynamicVar.GetType());

MyDynamicVar = "Hello World!!";
Console.WriteLine("Value: {0}, Type: {1}", MyDynamicVar,
MyDynamicVar.GetType());

MyDynamicVar = true;
Console.WriteLine("Value: {0}, Type: {1}", MyDynamicVar,
MyDynamicVar.GetType());

MyDynamicVar = DateTime.Now;
Console.WriteLine("Value: {0}, Type: {1}", MyDynamicVar,
MyDynamicVar.GetType());
```

Lesson : Set Date & Time

يوجد في C# Class : DateTime جاهز للتعامل مع التاريخ والوقت واسمه

```
//assigns default value 01/01/0001 00:00:00
DateTime dt1 = new DateTime();

//assigns year, month, day
DateTime dt2 = new DateTime(2023, 12, 31);

//assigns year, month, day, hour, min, seconds
DateTime dt3 = new DateTime(2023, 12, 31, 5, 10, 20);

//assigns year, month, day, hour, min, seconds, UTC timezone
DateTime dt4 = new DateTime(2023, 12, 31, 5, 10, 20, DateTimeKind.Utc);

Console.WriteLine(dt1);
Console.WriteLine(dt2);
Console.WriteLine(dt3);
Console.WriteLine(dt4);
```

Lesson : Get Current Datetime

الوقت الحالي

```
//assigns default value 01/01/0001 00:00:00
DateTime dt1 = new DateTime();

dt1 = DateTime.Now;
Console.WriteLine(dt1);
```

Lesson : Ticks

هو وحدة زمنية تمثل 100 نانو ثانية أي 10 مليون جزء من الثانية ❖

○ 100 نانو ثانية = 0.0000001 ثانية

❖ 1/1/0001 00:00:00.000 يبدأ من Datetime في Ticks ❖

Ticks

Ticks is a date and time expressed in the number of 100-nanosecond intervals that have elapsed since January 1, 0001, at 00:00:00.000 in the Gregorian calendar.

A single tick represents one hundred nanoseconds or one ten-millionth of a second.

There are 10,000 ticks in a millisecond and [10 million ticks in a second](#).

```
//number of 100-nanosecond intervals that have elapsed
//since January 1, 0001, at 00:00:00.000 in the Gregorian calendar.
```

```
DateTime dt = new DateTime();
Console.WriteLine(DateTime.MinValue.Ticks); //min value of ticks
Console.WriteLine(DateTime.MaxValue.Ticks); // max value of ticks
```

Lesson : DateTime Static Fields

بعض الـ Method في Datetime

```
DateTime currentDateTime = DateTime.Now; //returns current date and time
DateTime todaysDate = DateTime.Today; // returns today's date DateTime
currentDateTimeUTC = DateTime.UtcNow; // returns current UTC date and time
DateTime maxDateTimeValue = DateTime.MaxValue; // returns max value of DateTime
DateTime minDateTimeValue = DateTime.MinValue; // returns min value of DateTime

Console.WriteLine("currentDateTime: " + currentDateTime);
Console.WriteLine("Today: " + todaysDate);
Console.WriteLine("currentDateTimeUTC: " + currentDateTimeUTC);
Console.WriteLine("minDateTimeValue: " + minDateTimeValue);
Console.WriteLine("maxDateTimeValue: " + maxDateTimeValue);
```

Lesson : Time Span

و **DateTime** يمثل فترة زمنية وهو مصمم للعمل بسلامة مع **Object** لـ **Time Span** و **DateTimeOffset**

من استخدامات Time Span

- حساب الفرق بين تاريخين أو وقتين
- جدولة المهام
- الوقت المنقضي
- المقارنة بين تاريخين أو وقتين (`=>`, `=<`, `<`, `>`, `/`, `*`, `-`, `+`)

لا يمثل تاريخ بل فترة زمنية

```
DateTime dt = new DateTime(2023, 2, 21);

// Hours, Minutes, Seconds
TimeSpan ts = new TimeSpan(49, 25, 44);
Console.WriteLine(ts);
Console.WriteLine(ts.Days);
Console.WriteLine(ts.Hours);
Console.WriteLine(ts.Minutes);
Console.WriteLine(ts.Seconds);

//this will add time span to the date.
DateTime newDate = dt.Add(ts);

Console.WriteLine(newDate);
```

Lesson : Subtraction of two dates results in TimeSpan

من استخدامات Time Span حساب الفرق بين تاريخين أو وقتين

```
DateTime dt1 = new DateTime(2023, 2, 21);
DateTime dt2 = new DateTime(2023, 2, 25);
TimeSpan result = dt2.Subtract(dt1);

Console.WriteLine(result.Days);
```

Lesson : Operators

من استخدامات Time Span المقارنة بين تاريخين أو وقتين (= , < , > , / , * , - , +)

```
DateTime dt1 = new DateTime(2015, 12, 20);
DateTime dt2 = new DateTime(2016, 12, 31, 5, 10, 20);
TimeSpan time = new TimeSpan(10, 5, 25, 50);

Console.WriteLine(dt2 + time); // 1/10/2017 10:36:10 AM
Console.WriteLine(dt2 - dt1); //377.05:10:20
Console.WriteLine(dt1 == dt2); //False
Console.WriteLine(dt1 != dt2); //True
Console.WriteLine(dt1 > dt2); //False
Console.WriteLine(dt1 < dt2); //True
Console.WriteLine(dt1 >= dt2); //False
Console.WriteLine(dt1 <= dt2); //True
```

Lesson : Convert String to DateTime

تستطيع تحويل من التاريخ النصي الى DateTime باستخدام عدة Method

- throw an exception إذا كان هناك خطأ التنسيق سيحدث Parse() , ParseExact()
- bool والأفضل استخدامها لأنها من نوع TryParse() , TryParseExact()
- تعد في #C و & ++C في references إلى أدوات مشابهة لمرور المتغيرات ك Function

Convert String to DateTime

A valid date and time string can be converted to a object using Parse(), ParseExact(), TryParse() and TryParseExact() methods. **DateTime**

The Parse() and ParseExact() methods will throw an exception if the specified string is not a valid representation of a date and time. So, it's recommended to use TryParse() or TryParseExact() method because they return false if a string is not valid.

```
var str = "6/12/2023";
DateTime dt;

var isValidDate = DateTime.TryParse(str, out dt);

if (isValidDate)
    Console.WriteLine(dt);
else
    Console.WriteLine($"{str} is not a valid date string");

//invalid string date
var str2 = "6/65/2023";
DateTime dt2;

var isValidDate2 = DateTime.TryParse(str2, out dt2);

if (isValidDate2)
    Console.WriteLine(dt2);
else
    Console.WriteLine($"{str2} is not a valid date string");
```

Lesson : Quick Overview

C++ في String مثل C# في String

```
string S1 = "Mohammed Abu-Hadhoud";
Console.WriteLine(S1.Length);

//this will take 5 characters staring position 2
Console.WriteLine(S1.Substring(2, 5));
Console.WriteLine(S1.ToLower());
Console.WriteLine(S1.ToUpper());
Console.WriteLine(S1[2]);
Console.WriteLine(S1.Insert(3, "KKKK"));
Console.WriteLine(S1.Replace("m", "*"));
Console.WriteLine(S1.IndexOf("m"));
Console.WriteLine(S1.Contains("m"));
Console.WriteLine(S1.Contains("x"));
Console.WriteLine(S1.LastIndexOf("m"));

string S2 = "Ali,Ahmed,Khalid";
string[] NamesList = S2.Split(',');
Console.WriteLine(NamesList[0]);
Console.WriteLine(NamesList[1]);
Console.WriteLine(NamesList[2]);

string S3 = " Abu-Hadhoud ";
Console.WriteLine(S3.Trim());
Console.WriteLine(S3.TrimStart());
Console.WriteLine(S3.TrimEnd());
```

Lesson : String Interpolation

علامة \$ تدل على تنسيق النص (إبدال المتغير بالقيمة)⁵

```
// String Interpolation

string firstName = "Mohammed";
string lastName = "Abu-Hadhoud";
string code = "107";

//You shold use $ to $ to identify an interpolated string
string fullName = $"Mr. {firstName} {lastName}, Code: {code}";

Console.WriteLine(fullName);
```

Lesson : Casting Types : (Implicit Casting / Explicit Casting)

هي عملية تحويل قيمة من نوع بيانات الى نوع آخر
يوجد نوعان رئيسيان من التحويل

❖ التحويل الضمني **Implicit Casting**

- تحويل البيانات من النوع الأصغر int الى النوع الأكبر long
- يحدث تلقائيا من Compiler

❖ التحويل الصريح **Explicit Casting**

- تحويل البيانات من النوع الأكبر long الى النوع الأصغر int
- يحدث يدويا باستخدام (cost operator)

C# Type Casting

Type casting is when you assign a value of one data type to another type.

In C#, there are two types of casting:

- **Implicit Casting** (automatically) - converting a smaller type to a larger type size
`char` -> `int` -> `long` -> `float` -> `double`
- **Explicit Casting** (manually) - converting a larger type to a smaller size type
`double` -> `float` -> `long` -> `int` -> `char`

```
int age = 25;
double salary = 3500.50;

// (آمن)Implicit Casting
long largeNumber = age;

Console.WriteLine(age);
Console.WriteLine(largeNumber);

// (غير آمن) Explicit Casting
int truncatedSalary = (int)salary; // // سبودي إلى فقدان الجزء العشري

Console.WriteLine(salary);
Console.WriteLine(truncatedSalary);
```

Lesson : Type Conversion Methods

يوجد طرق أخرى للتحويل باستخدام **Convert(DataType)Method**

Type Conversion Methods

It is also possible to convert data types explicitly by using built-in methods, such

as,,,()

and `Convert.ToBoolean Convert.ToDouble Convert.ToString`
`Convert.ToInt32 int Convert.ToInt64 (long)`

```
int myInt = 20;
double myDouble = 7.25;
bool myBool = true;

Console.WriteLine(Convert.ToString(myInt));      // convert int to string
Console.WriteLine(Convert.ToDouble(myInt));       // convert int to double
Console.WriteLine(Convert.ToInt32(myDouble));     // convert double to int
Console.WriteLine(Convert.ToString(myBool));      // convert bool to string
```

Lesson : Casting Enums

تستطيع تحويل enum الى رقم ، وأيضاً تحويل الرقم الى enum

.Explicit casting is required to convert from an enum type to its underlying integral type

```
using System;
namespace Main
{
    internal class Program
    {
        enum WeekDays
        {
            Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday
        }
        static void Main(string[] args)
        {
            Console.WriteLine(WeekDays.Friday); //output: Friday
            int day = (int)WeekDays.Friday; // enum to int conversion
            Console.WriteLine(day); //output: 4

            var wd = (WeekDays)5; // int to enum conversion
            Console.WriteLine(wd); //output: Saturday
            Console.ReadKey();
        }
    }
}
```

Lesson : ReadLine();

لكن الاختلاف أن (cin >> : C++ مثلاً في Console.ReadLine() ترجع String ReadLine()

Get User Input

You have already learned that `is` is used to output (print) values, for input we use `Console.ReadLine()` `Console.WriteLine()`

Equivalent to `cin>>` in C++

Important: `Console.ReadLine()` always reads string.

```
// Type your username and press enter
Console.WriteLine("Enter username?");

string userName = Console.ReadLine();
Console.WriteLine("Username is: " + userName);
```

Lesson : User Input and Numbers

عندما تريد قراءة مدخلات من المستخدم من نوع آخر غير `string` **لابد من تحويل البيانات من Convert.DataType(ReadLine());** `string` **إلى النوع الذي تريده**

User Input and Numbers

The `method` returns a `.string`. Therefore, you cannot get information from another data type, such as `Console.ReadLine()` `string int`

therefore you should use casting when you read.

```
Console.WriteLine("Enter your age?");
//if you don't convert you will get error and if you enter string you will get error
int age = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("Your age is: " + age);
```

Lesson : (Assignment / Arithmetic / Relational / Logical) Operators

int x = 10; تخزين القيم في المتغيرات : Assignment Operators ♦♦

int x = 5 ; int y = 10; العمليات الحسابية مثل : Arithmetic Operators ♦♦

Int z = x + y ; // z = 15 ○

Int z = y - x ; // z = 5 ○

Int z = x * y ; // z = 50 ○

Int z = y / x ; // z = 2 ○

Int z = y % x ; // z = 0 ○

(true / false) bool r ; عمليات المقارنة : Relational Operators ♦♦

r = x == y ; // r = false ○

r = x > y ; // r = false ○

r = x < y ; // r = true ○

r = x >= y ; // r = false ○

r = x <= y ; // r = true ○

r = x != y ; // r = true ○

(AND = && / OR = || / NOT = !) العمليات المنطقية : Logical Operators ♦♦

true && true = true ○

true && false = false / false && true = false / false && false = false ○

false || false = false ○

true || false = true / false || true = true / true || true = true ○

true != true = false ○

true != false = true / false != true = true / false != false = true ○

تماثل مع C++

C# Logical operators			
Operand1	Operand 2	OR ()	AND (&&)
true	true	true	true
true	false	true	false
false	true	true	false
false	false	false	false

C# Compound Assignment Operators			
Operator	Operator Name	Example	Equivalent To
<code>+=</code>	Addition Assignment	<code>x += 5</code>	<code>x = x + 5</code>
<code>-=</code>	Subtraction Assignment	<code>x -= 5</code>	<code>x = x - 5</code>
<code>*=</code>	Multiplication Assignment	<code>x *= 5</code>	<code>x = x * 5</code>
<code>/=</code>	Division Assignment	<code>x /= 5</code>	<code>x = x / 5</code>
<code>%=</code>	Modulo Assignment	<code>x %= 5</code>	<code>x = x % 5</code>
<code>&=</code>	Bitwise AND Assignment	<code>x &= 5</code>	<code>x = x & 5</code>
<code> =</code>	Bitwise OR Assignment	<code>x = 5</code>	<code>x = x 5</code>
<code>^=</code>	Bitwise XOR Assignment	<code>x ^= 5</code>	<code>x = x ^ 5</code>
<code><<=</code>	Left Shift Assignment	<code>x <<= 5</code>	<code>x = x << 5</code>
<code>>>=</code>	Right Shift Assignment	<code>x >>= 5</code>	<code>x = x >> 5</code>
<code>=></code>	Lambda Operator	<code>x => x*x</code>	Returns <code>x*x</code>

C# Arithmetic Operators		
Operator	Operator Name	Example
<code>+</code>	Addition Operator	<code>6 + 3 evaluates to 9</code>
<code>-</code>	Subtraction Operator	<code>10 - 6 evaluates to 4</code>
<code>*</code>	Multiplication Operator	<code>4 * 2 evaluates to 8</code>
<code>/</code>	Division Operator	<code>10 / 5 evaluates to 2</code>
<code>%</code>	Modulo Operator (Remainder)	<code>16 % 3 evaluates to 1</code>

C# Relational Operators		
Operator	Operator Name	Example
<code>==</code>	Equal to	<code>6 == 4 evaluates to false</code>
<code>></code>	Greater than	<code>3 > -1 evaluates to true</code>
<code><</code>	Less than	<code>5 < 3 evaluates to false</code>
<code>>=</code>	Greater than or equal to	<code>4 >= 4 evaluates to true</code>
<code><=</code>	Less than or equal to	<code>5 <= 3 evaluates to false</code>
<code>!=</code>	Not equal to	<code>10 != 2 evaluates to true</code>

Lesson : Unary Operators / Ternary Operator

C# unary operators		
Operator	Operator Name	Description
+	Unary Plus	Leaves the sign of operand as it is
-	Unary Minus	Inverts the sign of operand
++	Increment	Increment value by 1
--	Decrement	Decrement value by 1
!	Logical Negation (Not)	Inverts the value of a boolean

```
int number = 10, result;
bool flag = true;

result = +number;
Console.WriteLine("+number = " + result);

result = -number;
Console.WriteLine("-number = " + result);

result = ++number;
Console.WriteLine("++number = " + result);

result = --number;
Console.WriteLine("--number = " + result);

Console.WriteLine("!flag = " + (!flag));

Console.WriteLine((number++)); // 10
Console.WriteLine((number)); // 11

Console.WriteLine((++number)); // 12
Console.WriteLine((number)); // 12
```

⁶ C++ هو اختصار كتابة الشرط مثل في **Ternary Operator** ♦♦

```
int number = 12;
string result;

result = (number % 2 == 0) ? "Even Number" : "Odd Number";
Console.WriteLine("{0} is {1}", number, result);
```

Bitwise and Bit Shift Operators ♦

Bitwise Complement (~ tilde) ○

يعكس كل bit – في Binary – من 0 الى 1 ومن 1 الى 0 ■

```
int number = 10; // 1010 binary
int complement = ~number; // 0101 (-11) binary
```

فهي شائي
number = 5; // 0101
complement = ~number; // 1010 (-6) binary

Bitwise AND (&) ○

المقارنة بين 2 : إذا كان كلاهما 1 ف bit = 1 إذا اختلفوا = 0 ■

```
int x = 10; // 1010 binary
int y = 6; // 0110 binary
int result = x & y; // 0010 (2) binary
```

Bitwise OR (|) ○

المقارنة بين 2 : إذا كان أحدهما 1 ف bit = 1 إذا كانوا 0 = 0 ■

```
int x = 10; // 1010 binary
int y = 6; // 0110 binary
int result = x | y; // 1110 (14) binary
```

Bitwise Exclusive OR – XOR – (^) ○

المقارنة بين 2 : إذا اتفقا في 1 أو 0 ف bit = 0 إذا اختلفوا = 1 ■

```
int x = 10; // 1010 binary
int y = 6; // 0110 binary
int result = x ^ y; // 1100 (12) binary
```

Bitwise Left Shift ○

Bitwise Right Shift ○

نقل – 0 – إما لبداية (>>=) binary أو لنهاية (<<=) binary بحسب عدد

مرات الإزاحة Shift ■

```
int number = 5; // 101 binary
// إزاحة bitwise الى اليسار
number <<= 2; // 10100 (20) binary
// إزاحة bitwise الى اليمين
number >>= 1; // 1010 (10) binary
```

Lesson : if, if...else, if...else if and Nested if Statement

C++ مثله في if else⁷

```
//if else if statement
int number = 12;

if (number < 5)
{
    Console.WriteLine("{0} is less than 5", number);
}
else if (number > 5)
{
    Console.WriteLine("{0} is greater than 5", number);
}
else
{
    Console.WriteLine("{0} is equal to 5");
}
```

Lesson : switch Statement / ternary (?) Operator / for loop

string مثله في C# ولكن switch في C++ لا تدعم string أما في C++ مثله في switch⁸ ❖
لابد من استخدام break في نهاية كل حالة : للخروج من switch ○

C++ اختصار if else⁹ternary (?) Operator ❖
Condition ? Expression1 : Expression2; ○

C++ مثله في for loop¹⁰ ❖

```
for (initialization; condition; iterator)
{
    // body of for loop
}
```

⁷ الكورس 3 الدرس 43 & 42

⁸ الكورس 3 الدرس 44

⁹ الكورس 6 الدرس 9

¹⁰ الكورس 3 الدرس 47

Lesson : while loop / do while loop / (break / continue) Statement

C++ – مثله في syntax – في while loop¹¹ ♦♦♦

```
while (test-expression)
{
    // body of while
}
```

C++ – مثله في syntax – في do while loop¹² ♦♦♦

```
do
{
    // body of do while loop
} while (test-expression);
```

break Statement¹³ تستخدم للتوقف أو الخروج من حلقة التكرار ♦♦♦

for loop / if else / while loop / switch ○

continue Statement¹⁴ تستخدم للرجوع الى بداية loop وعدم تنفيذ Code الذي بعدها ♦♦♦

for loop / if else / while loop ○

¹¹ الكورس 3 الدرس 50

¹² الكورس 3 الدرس 51

¹³ الكورس 3 الدرس 52

¹⁴ الكورس 3 الدرس 54

Lesson : Arrays Are Bound / Array Declaration

❖ في C# لها حجم ثابت يتم تعريفه عند إنشائها ، لا يمكن تغيير حجم Array بعد إنشائها ومن فوائدها الأمان والكفاءة والبساطة في استخدامها بخلاف C++ التي يمكن تغيير حجمها بعد إنشائها

```
// create an array
int[] numbers = { 1, 2, 3 };

numbers[0] = 1;
numbers[1] = 2;
numbers[2] = 3;

numbers[3] = 4; ✘

Console.ReadKey()
}

Exception Unhandled
System.IndexOutOfRangeException: 'Index was outside the bounds
of the array.'

Show Call Stack | View Details | Copy Details | Start Live Share session
↳ Exception Settings
```

❖ عند إنشاء Array تستخدم معها new

```
int[] age = new int[5];
```

C# Array Declaration

In C#, here is how we can declare an array.

```
datatype[] arrayName;
```

Here,

- **datatype** - data type like,,, etc `int string char`
- **arrayName** - it is an identifier

Let's see an example,

```
int[] age;
```

Here, we have created an array named age. It can store elements of type. `int`

But how many elements can it store?

To define the number of elements that an array can hold, we have to allocate memory for the array in C#. For example,

```
// declare an array
int[] age;

// allocate memory for array
age = new int[5];
```

Here, represents that the array can store 5 elements. We can also say the size/length of the array is 5. `new int[5]`

Note: We can also declare and allocate the memory of an array in a single line. For example,

```
int[] age = new int[5];
```

Lesson : Array initialization / Access Array Elements

يوجد أكثر من طريقة لإنشاء **Array**

الطريقة الأولى تحديد عدد العناصر

```
// declare an array  
int[] age = new int[5];  
  
//initializing array  
age[0] = 12;  
age[1] = 4;  
age[2] = 5;
```

الطريقة الثانية : إضافة القيم بين {}, و Compiler يحسب عدد العناصر بشكل تلقائي

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

تستطيع الوصول الى العناصر عن طريق **Index**

≡ **Array initialization**

In C#, we can initialize an array during the declaration. For example,

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

Here, we have created an array named numbers and initialized it with values 1, 2, 3, 4, and 5 inside the curly braces.

Note that we have not provided the size of the array. In this case, the C# automatically specifies the size by counting the number of elements in the array (i.e. 5).

In an array, we use an **index number** to determine the position of each array element. We can use the index number to initialize an array in C#. For example,

```
// declare an array  
int[] age = new int[5];  
  
//initializing array  
age[0] = 12;  
age[1] = 4;  
age[2] = 5;  
...
```

Note:

- An array index always starts at 0. That is, the first element of an array is at index 0.
- If the size of an array is 5, the index of the last element will be at 4 (5 - 1).

Lesson : Two-Dimensional Array

تستطيع إنشاء Two-Dimensional Array عن طريق [,] Name Datatype

```
//initializing 2D array
int[,] numbers = { { 12, 13 }, { 55, 77 } };

// access first element from the first row
Console.WriteLine("Element at index [0, 0] : " + numbers[0, 0]);

// access first element from second row
Console.WriteLine("Element at index [1, 0] : " + numbers[1, 0]);
```

Two-Dimensional Array Declaration

Here's how we declare a 2D array in C#.

```
int[,] x = new int[2, 3];
```

Here, x is a two-dimensional array with 2 elements. And, each element is also an array with 3 elements.

So, all together the array can store 6 elements (2 * 3).

Note: The single comma [,] represents the array is 2 dimensional.

Two-Dimensional Array initialization

In C#, we can initialize an array during the declaration. For example,

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

Here, x is a 2D array with two elements and . We can see that each element of the array is also an array. {1, 2, 3} {3, 4, 5}

We can also specify the number of rows and columns during the initialization. For example,

```
int[,] x = new int[2, 3] { { 1, 2, 3 }, { 3, 4, 5 } };
```

Lesson : for each loop

for each loop هو لكرار مجموعة من البيانات مثل ... List , Array

```
char[] myArray =
{ 'H', 'e', 'l', 'l', 'o' };

foreach (char ch in myArray)
{
    Console.WriteLine(ch);
}
```

C# foreach loop

We will learn about foreach loops (an alternative to for loop) and how to use them with arrays and collections.

C# provides an easy to use and more readable alternative to for loop, the foreach loop when working with arrays and collections to iterate through the items of arrays/collections. The foreach loop iterates through each item, hence called foreach loop.

Syntax of foreach loop

```
foreach (element in iterable-item)
{
    // body of foreach loop
}
```

Here iterable-item can be an array or a class of collection.

Lesson : Array Operations using System.Linq

Max / Min / Count / Sum / Average مثل Array هي مكتبة لإجراء العمليات في **System.Linq**

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadoud

using System;

// provides us various methods to use in an array
using System.Linq;

namespace Main
{
    internal class Program
    {

        static void Main(string[] args)
        {

            // Note that we used System.Linq above.

            int[] numbers = { 51, -1, 2, 14, 18, 40, 178 };

            // get the minimum element
            Console.WriteLine("Smallest Element: " + numbers.Min());

            // Max() returns the largest number in array
            Console.WriteLine("Largest Element: " + numbers.Max());

            // compute Count
            Console.WriteLine("Count : " + numbers.Count());

            // compute Sum
            Console.WriteLine("Sum : " + numbers.Sum());

            // compute the average
            Console.WriteLine("Average: " + numbers.Average());

            Console.ReadKey();
        }
    }
}
```

Lesson : C# Math

مكتبة **Math** فيها العديد من العمليات الرياضية

```
Console.WriteLine("Max of 5, 10 is: {0}", Math.Max(5, 10));
Console.WriteLine("Min of 5, 10 is: {0}", Math.Min(5, 10));
Console.WriteLine("Squar Root of 64 is: {0}", Math.Sqrt(64));
Console.WriteLine("Absolute (positive) value of -4.7 is: {0}",
Math.Abs(-4.7));
Console.WriteLine("Round of 9.99 is: {0}", Math.Round(9.99));
```

Lesson : C# Methods

C# Methods

A **method** is a block of code which only runs when it is called.

//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

```
using System;
```

You can pass data, known as parameters, into a method.

```
namespace Main
{
    internal class Program
    {
```

Methods are used to perform certain actions, and they are also known as **functions**.

C# is a fully OOP language , you cannot create a method outside class.

Create a Method

A method is defined with the name of the method, followed by parentheses (). C# provides some pre-defined methods, which you already are familiar with, such as , but you can also create your own methods to perform certain actions: **Main()**

Example:

```
static void MyMethod()
{
    // code to be executed
}
```

You should use **static** if you want to call the method without having object.

```
static void PrintMyName()
{
    Console.WriteLine("Mohammed Abu-Hadhoud");
}
static void Main(string[] args)
{

    PrintMyName();
    Console.ReadKey();
}
```

Lesson : Method Parameters / Default Parameter Value

C# Method Parameters

Parameters and Arguments

Information can be passed to methods as parameter. Parameters act as variables inside the method.

They are specified after the method name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma.

C# Default Parameter Value

Default Parameter Value

You can also use a default parameter value, by using the equals sign 0. =

```
using System;

namespace Main
{
    internal class Program
    {

        static void PrintMyInfo(string Name, byte Age)
        {
            Console.WriteLine("Name= {0} , Age= {1}", Name, Age);
        }

        static void PrintMyInfo(string Name, byte Age, string Address = "No
Address")
        {
            Console.WriteLine("Name= {0} , Age= {1}, Address= {2}", Name, Age,
Address);
        }

        static void Main(string[] args)
        {

            PrintMyInfo("Mohammed-AbuHadhoud", 45);

            //First we did not provide the address it's optional
            PrintMyInfo("Mohammed-AbuHadhoud", 45);

            //second we provided the address
            PrintMyInfo("Mohammed-AbuHadhoud", 45, "Amman-Jordan");

            Console.ReadKey();
        }
    }
}
```

Lesson : Return Values / Named Arguments

C# Return Values

Return Values

In the previous lessons, we used the `void` keyword in all examples, which indicates that the method should not return a value.

If you want the method to return a value, you can use a primitive data type (such as `int` or `double`) instead of `void`, and use the `return` keyword inside the method.

C# Named Arguments

Named Arguments

It is also possible to send arguments with the syntax `key: value`

That way, the order of the arguments does not matter.

```
using System;
namespace Main
{
    internal class Program
    {

        static string GetMyName()
        {
            return "Mohammed-AbuHadhoud";
        }

        static void MyMethod(string child1, string child2, string child3)
        {
            Console.WriteLine("The youngest child is: " + child3);
        }

        static void Main(string[] args)
        {

            Console.WriteLine("My Name is {0}", GetMyName());
            //see the order of sending parameters is not important.

            MyMethod(child3: "Omar", child1: "Saqer", child2: "Hamza");
            Console.ReadKey();
        }
    }
}
```

ليس شرط أن تكتب **Parameters** مرتبة كما في **Function** بل يمكنك كتابة اسمه ثم : ثم القيمة
(Key : value)

Lesson : Method Overloading

Method Overloading

With **method overloading**, multiple methods can have the same name with different parameters.

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadoud

using System;

namespace Main
{
    internal class Program
    {

        static int Sum(int Num1, int Num2)
        {
            return Num1 + Num2;
        }

        static int Sum(int Num1, int Num2, int Num3)
        {
            return Num1 + Num2 + Num3;
        }

        static int Sum(int Num1, int Num2, int Num3, int Num4)
        {
            return Num1 + Num2 + Num3 + Num4;
        }

        static void Main(string[] args)
        {
            //we have 3 different methods but with the same name.

            Console.WriteLine(Sum(10, 20));
            Console.WriteLine(Sum(10, 20, 30));
            Console.WriteLine(Sum(10, 20, 30, 40));
            Console.ReadKey();
        }
    }
}
```

Lesson : Exceptions / Random Function In C#

١٥: عند تنفيذ الكود قد تحصل بعض الأخطاء مثل الإدخال الخاطئ Exception Handling

C# Exceptions - Try..Catch

C# Exceptions

When executing C# code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things.

When an error occurs, C# will normally stop and generate an error message. The technical term for this is: C# will throw an **exception** (throw an error).

C# try and catch

The **try** statement allows you to define a block of code to be tested for errors while it is being executed. **try**

The **catch** statement allows you to define a block of code to be executed, if an error occurs in the try block. **catch**

The **try** and **catch** keywords come in pairs: **try catch**

Syntax:

```
try
{
    // Block of code to try
}
catch (Exception e)
{
    // Block of code to handle errors
}
```

```
try
{
    int[] myNumbers = { 1, 2, 3 };
    Console.WriteLine(myNumbers[10]);
}
catch (Exception e)
{
    Console.WriteLine(e.Message);
}
```

الحصول على رقم عشوائي ضمن نطاق محدد : Random Function In C#

```
Random rnd = new Random();

for (int j = 0; j < 4; j++)
{
    Console.WriteLine(rnd.Next(10, 20));
    // returns random integers >= 10 and < 20
}
```

Generate Random Number in Min to Max Range

Use the `Next` overload method to get a random integer that is within a specified range. `Next(int min, int max)`

Example: Generate Random Integers in Range Copy

```
Random rnd = new Random();

for(int j = 0; j < 4; j++)
{
    Console.WriteLine(rnd.Next(10, 20)); // returns random integers >= 10 and < 20
}
```

Syntax C# vs C++

C++	C#
#include <iostream>	using System;
int main()	static void Main(string[] args)
cout << " C++ \n " ;	Console.WriteLine (" C# ");
system("pause>0");	Console.ReadLine();
cout << "The sum of 10 + 20 is " << 10 + 20;	Console.WriteLine("The sum of 10 + 20 is " + (10 + 20));
cout << " C++ " ;	Console.Write("C# ");
char x[] = "C++" ; Printf(" %s " , x);	Console.Write(" {0} " , "C# ");
\n , \t , \' , \" , \\ , \a	\n , \t , \' , \" , \\ , \a
تحذف الحرف الذي قبلها	\b
(/**/ , //) Comments	(/* */ , //) Comments
(int, string, char, bool) Name = value	(int, string, char, bool) Name = value
double Name = value (12.445)	double Name = value D (12.445D)
✗ @break	✓ @break
auto x = 10.5;	var x = 10.5;
short هي Datatype أصغر	byte هي Datatype أصغر
enum EN { };	enum EN { };
struct Name { int x ; };	struct Name { public int x ; }
&	out
string S1 = "C++";	string S1 = "C#";

<code>cin >> ; // whatever</code>	<code>Console.ReadLine(); // string</code>
<code>cin >> ; // whatever</code>	<code>Convert.DataType(ReadLine());</code>
<code>if (1 < 2) { ... } else if (...) { ... } else { .. }</code>	<code>if (1 < 2) { ... } else if (...) { ... } else { .. }</code>
<code>Switch("string") { case "Saeed" : // code ; break;} ❌</code>	<code>Switch("string") { case "Saeed" : // code ; break;} ✅</code>
<code>Condition ? Expression1 : Expression2;</code>	<code>Condition ? Expression1 : Expression2;</code>
<code>for (initialization; condition; iterator) { ... }</code>	<code>for (initialization; condition; iterator) { ... }</code>
<code>while (Condition) { // code }</code>	<code>while (Condition) { // code }</code>
<code>do { // code } while(Condition);</code>	<code>do { // code } while(Condition);</code>
<code>int arr[8];</code>	<code>int[] arr = new int[5]</code>
<code>int tow [3][4] ;</code>	<code>int[,] tow = new int [3 , 4];</code>
<code>for (int i = 0 ; i < 5 ; i++) { ... }</code>	<code>foreach (int x in arr) { ... }</code>

Desktop Applications تستخدم Windows Forms وعملها تطوير برامج Desktop Applications يكون موجود في Server و تستطيع الوصول إليه عن طريق Browser Web Applications هو برنامج يتم تزيله على الجهاز مثل برامح Office المميزات والمزايا أو من الأفضل Web أم Desktop تعتمد على ما تحتاجه – أو حسب مميزات البرنامج أو احتياج الزبون أو سعر البرنامج –

من مميزات Desktop Applications

١. تستطيع الوصول إلى التطبيق من غير إنترنت Offline مثل Visual Studio
٢. فيها خصوصية أكثر للمستخدم لذا هو أكثر أمان
٣. التكلفة تكون قليلة – تزيل وتنصب البرنامج على الجهاز –
٤. أسرع من Web لأنها مثبتة في الجهاز وهو Offline ويستخدم كل موارد الجهاز Desktop Adobe يوجد بعض الأمور لا توجد إلا في Desktop مثل ... RAM , CPU

من مساوى Desktop Applications

١. لابد من تثبيت البرنامج على الجهاز – أخذ مساحة من الذاكرة وجهاز ذو مواصفات جيدة –
٢. التحديث ليس تلقائي بل يدوى

في Web يكون التحديث تلقائي في Web لا تحتاج جهاز ذو مواصفات جيدة بل متصفح Browser في الإنترت يكون كل شيء مسجل أو مراقب لذا هو أقل خصوصية وأمان التكلفة عالية مثل Server وتحديثات وغيرها مثل منصة Programming Advices سرعته أقل من Desktop لأنه يعتمد على سرعة الإنترت وعدد المستخدمين في Web بشكل عام أسرع من Desktop Web

Web vs Desktop Applications



Web-based applications are programs that you access online. The function is delivered to your device from a remote server when you access it via your browser— it's not installed on your machine.

Desktop applications, are the programs that you download and install on your device. Everything they need to deliver your functionality is stored on your workstation. So, you can access the application through your desktop.

Examples: Word, Excel , Browsers, Photoshop...etc.

Pros (Advantages) of Desktop Applications

Offline Access



Internet access is not necessary while working with a desktop application, as it is installed in the system and you are leveraged to access it any time.

Still, there can be a situation when a person cannot find a stable connection. However, a desktop app will not impact its work, and all the procedures will be executed successfully, providing the required output.

Furthermore, once the application is installed, you can use it anywhere, anytime.

More Privacy



In terms of data security, desktop solutions are considered more precisely, as overall data is stored on your machine. Therefore, you are the only authorized person to access, read, write and share it.

In addition, whenever someone works on a web-based app, cookies are generated, browser history is saved, enabling the companies to display ads.

Moreover, your device will be secure from man-in-the-middle and other threats in offline mode, and auto-downloading of any malware will also be prevented.

Cost-Efficient



Applications based on Software as a Service are pretty expensive than desktop solutions, as the vendor has to host and maintain it constantly.

Also, you have to pay hefty amounts for accessing the services through a web app. On the other hand, most desktop-based applications are freely available, and if any one of them needs to be purchased, the user can use it for a long-time.

In addition, if an enterprise provides its desktop application to its stakeholders, it only has to host a web page with a link, and the company can do it in a small amount. In this way, desktop apps help both end-users and organizations to save money.

Optimized Performance



If we compare the web and desktop apps based on speed, desktop applications always will be the dominant technology.

It directly links with the device and utilizes all the required resources appropriately. Users do not generally analyze command execution, but it is faster in desktop apps, as low bandwidth doesn't impact the performance.

Besides this, a wide array of features is offered compared to web solutions, as computer systems can handle many threads simultaneously. For instance, you can manually allocate the space to software to maximize its functioning.

Cons (Disadvantages) of Desktop Applications Installation is necessary



While using a web-based application, a browser and internet access are the only basic requirements.

But, to use a desktop app, installing it on the device is a must. And, for the successful installation, aspects such as Operating system, RAM, processor, GUI, ROM, and many more are analyzed.

Therefore, it makes the desktop app a little a bit expensive. Moreover, it is difficult to switch between systems to complete the work, as it is native to a single OS and can't be accessed on any other system.

More Space Utilization



Desktop Software requires larger storage space on the machine's hard disk to deploy its necessary packages and associated files.

Also, any new file the user adds is stored on the system, utilizing more space, which loads the system.

In addition, the cost of maintenance can be increased with this if there is insufficient space on the hard disk installed on the device, as the user needs to modify the disk with a larger capacity.

Upgraded Manually



Downloading a new file whenever an update is available is sometimes a hectic piece of work, as you have to wait for its compatibility check, installation, and then rebooting the system for successful configuration.

In addition, it also consumes more space and processing cores, which can degrade the performance due to a shortage of resources.

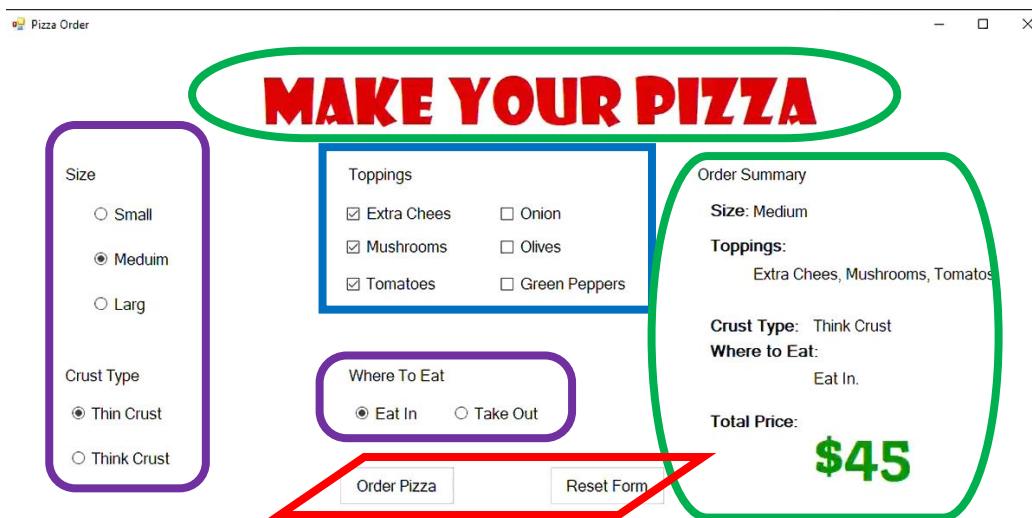
Wrapping Up



Web Application and Desktop Application both are heavily selected from small to large scale organizations, as per their need and budget.

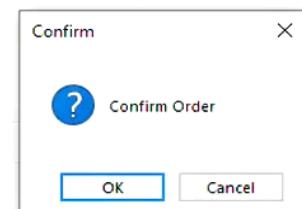
Both these solutions have their advantages and disadvantages.

However, desktop-based software is being considered for internal use in enterprises, and web-based apps are mostly preferred for users at remote locations. It depends upon your requirements.



الشاشة التي تظهر للمستخدم تسمى **Form** يعني نموذج ويوجد بداخله العديد من **Controls** و **Controls** له أنواع وكل نوع له خاصته – ستدرس لاحقاً – منها

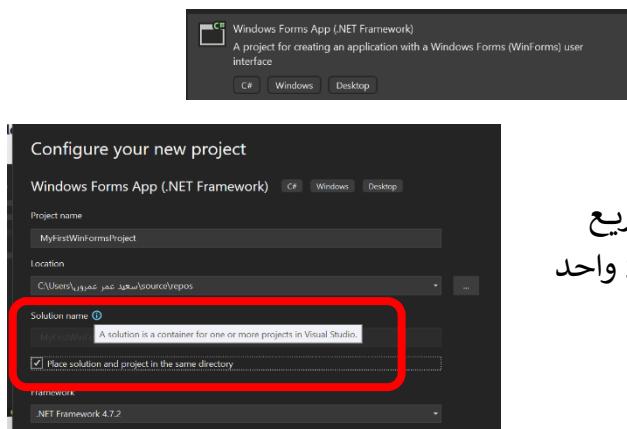
- اختيار واحد من عدة اختيارات – **Radio Button** •
- تستطيع اختيار عدة خيارات أو لا تختار شيء - **Check box Control** •
- عنوان لا تستطيع الكتابة عليه مباشرة ○ تستطيع تغيير حجم ولون العرض أو مواصفاته
- أزرار – الخيارات ل كامل البرنامج **Button** •
- رسالة تأكيد الخيارات وعدم تحديث الخيارات **Massage box** •



ويوجد فيها شيئين **Control & Properties** ووظيفته عند اختيار **Control** معين يتم مناداة **Function** معين ويسمى **On Click**

Visual Class هو **Control**

Lesson 33 : Part 2 - First Windows Forms Application



يعمل تطبيق Desktop على إطار

عند اختيار Solution تستطيع إضافة عدة مشاريع منفصلة عن بعضها وربطها معا تحت Solution واحد بخلاف Project

عند فتح المشروع تظهر شاشة Form التي تستطيع من خلالها تصميم Controls
عند تشغيل البرنامج تظهر شاشة Form ل Run Time - شاشة النتائج -

عند Design تصميم البرنامج تحتاج الى أدوات تساعدك على العمل وهي Toolbox وتستطيع الوصول إليها : View >> Toolbox او باختصار Ctrl + Alt + X

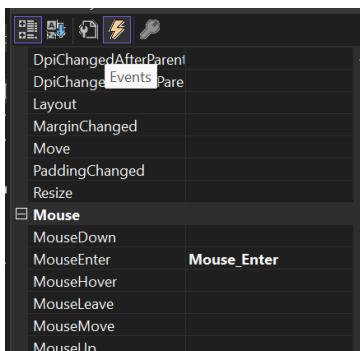
فيها عدة أدوات ضمن مجموعات ،
ومنها مجموعة شائعة الاستخدام وهي Common Controls ومن أدواتها

- Textbox وهو مكان لاستقبال المدخلات من المستخدم عادة
- Button وهو زر لتنفيذ كود معين عند استخدامه

عند استخدام Control معين يتم سحبه الى Form وتصميم حجمه

وللذهاب الى مواصفات Control المراد في Form الرئيسية - ضغطة F4 - ثم فتظهر شاشة فيها كل خصائص Control الخاصة بـ Properties ومن هذه الخصائص

Form هو الاسم الذي يظهر للمستخدم في Text
(Name) هو الاسم الذي تتعامل معه في الكود - وهذا Name تستطيع من خلاله الوصول الى جميع Properties لـ Control في الكود - يشبه Object وبداخله Members



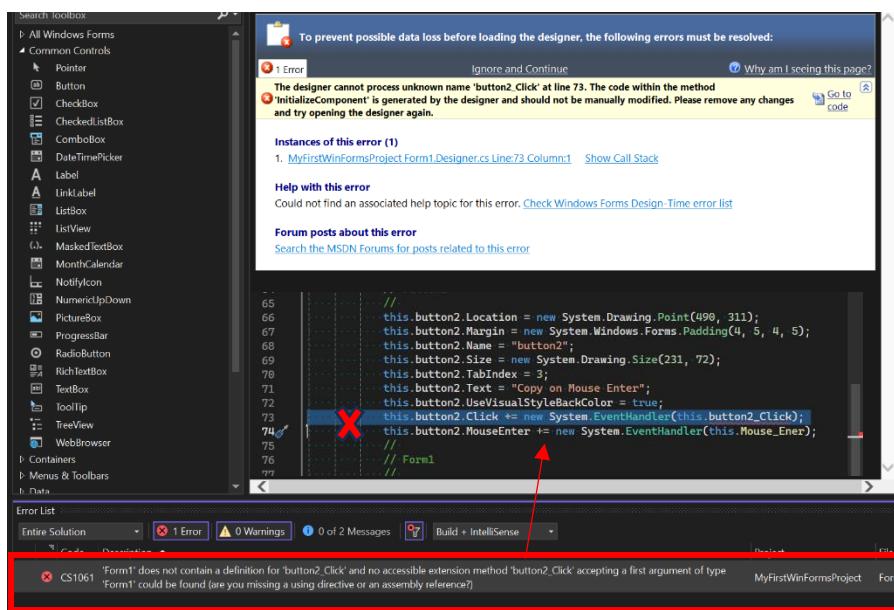
وللذهاب الى الكود لـ Control المراد تحده - تضغط F2 - فيذهب الى Function وتنكتب الكود

ولإنشاء مؤثرات على Button تذهب الى Events وهي حدث Function يحصل لـ Control فینادي Function

Lesson 34 : Introduction About Controls – Part3

مكون من شيئين **Design** وال**كود**
 تستطيع إعادة فتحهما عن طريق **لو تم إغلاقهما**

F7 تضغط **Code behind** ولإظهار شاشة **Design** فتظهر شاشة **View >> Solution Explorer**



عند حدوث مثل هذه الأخطاء : فتضغط 2 فيذهب بك الى مكان الخطأ في الكود فتحذفه

بعض الخصائص Control Properties



- لجعل Control لا يتم التعديل عليه – للقراءة فقط –
- لإخفاء Control من شاشة المستخدم
- لتغيير لون الخلفية ل Control
- لتغيير حجم أو اللون أو الخط
- ❖ هناك تكامل في Event بين Controls مثل
- Text1 & Text2 – Control
- ❖ تستطيع أيضا الوصول الى Form الرئيسية إما عن طريق **this** في الكود أو F4
- لتغيير لون الخلفية ل **Form . Back Color**
- تغيير العنوان **Form . Text** وغيرها
- ❖ تستطيع تغيير خصائص Control من شاشة Properties أو عن طريق اسم Control ثم . ثم كتابة عنوان **Label : Controls Common Controls**

Lesson 35 : Part 4 : Some formatting and Alignments



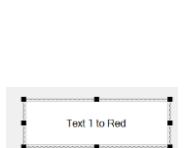
تستطيع تغيير حجم وحدود الشكل ل **Control** عن طريق
ثم التعديل على الباقي Flat Style >> Flat



- لتغيير لون الشكل عند الاستمرار على الضغط عليه
- لتغيير لون الشكل عند المرور فوق الشكل
- لتغيير لون حدود الشكل
- لتغيير حجم حدود الشكل

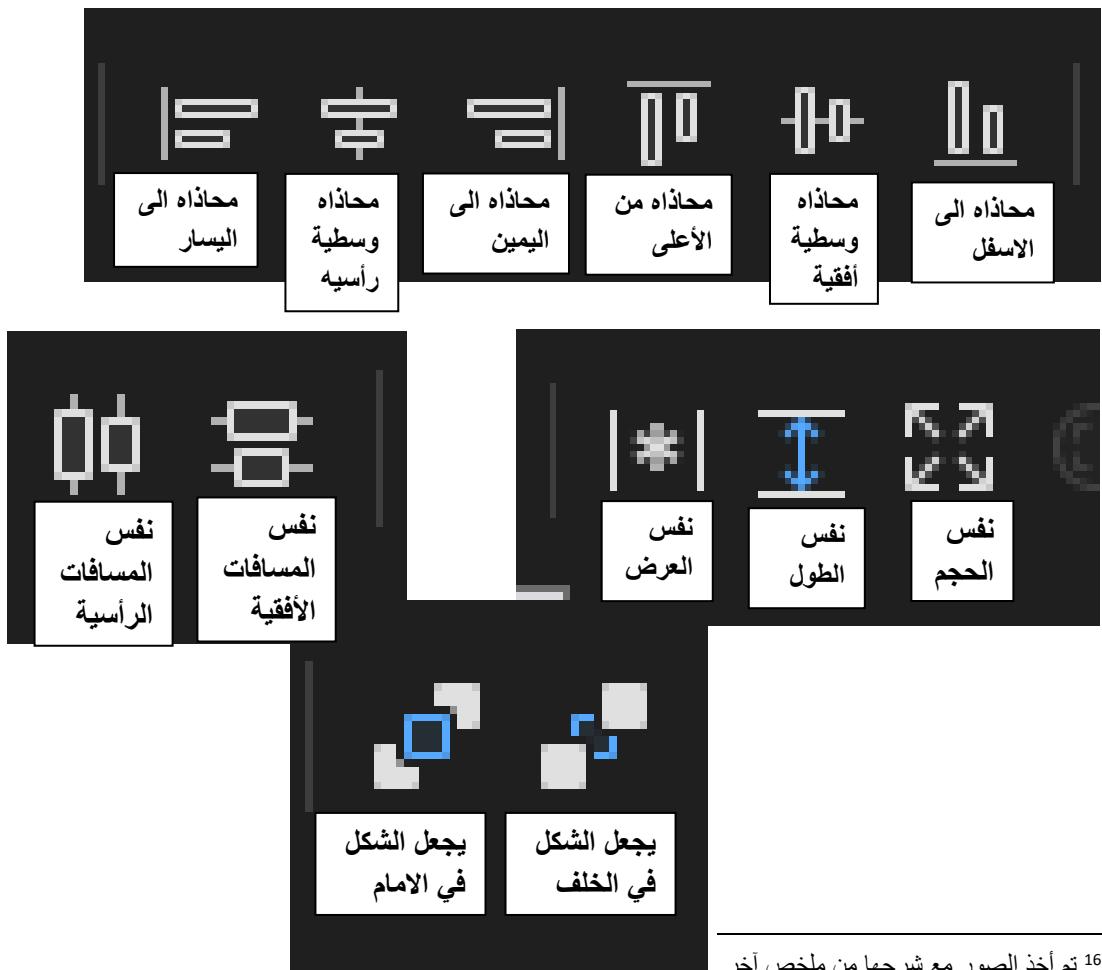
تأتي قائمة الألوان System من الصورة في الجهاز مثل الألوان من الثيمات

لترتيب **Control** على نسق موحد دفعة واحدة



- ❖ تحديد أولاً الشكل الرئيسي تكون المربعات بيضاء
- ❖ ثم تحديد الأشكال التابعة له عن طريق (الشكل المراد + Ctrl)

أدوات تنسيق Controls ¹⁶ في Design



¹⁶ تم أخذ الصور مع شرحها من ملخص آخر

Lesson 36 : Part 5 : TabIndex, TabStop, and New Forms (Default, Show, ShowDialog, Close)

شرط مهم في تطبيقات Desktop : العمل على التطبيق من غير ماوس

لترتيب Tab للمور على Controls بشكل مرتب : يوجد في كل Control Property TabStop و TabIndex اسمها

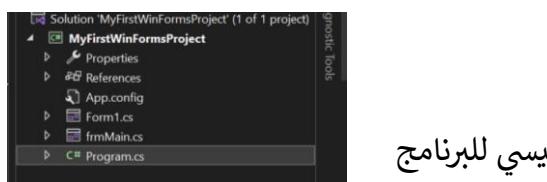


لإضافة Form جديد

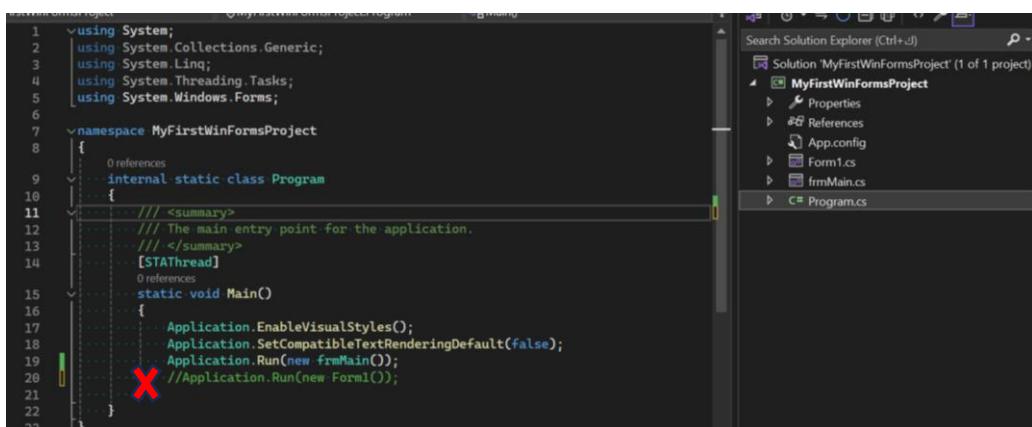
زر الفأرة الأيمن على Solution ثم Add ثم Form (Windows Forms)
أو باختصار : (Ctrl + Shift + A) ثم



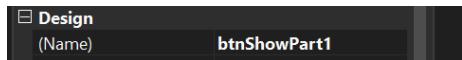
لتغيير اسم Form من Form ثم Solution ثم Form المراد بزر الفأرة الأليم ثم Rename أو حدد Form ثم F2



عند تشغيل البرنامج يتم تشغيل Form1 بشكل تلقائي ،
لتتشغيل frmMain : تذهب الى Solution ثم الى الكود الرئيسي للبرنامج
frmMain الى Form1 : تبدل frmMain الى



قاعدة في تسمية Control – الاسم الذي يتعامل معه في الكود (Name)



- يكون بداية اسمه : **btnName** ماذا يعمل Button
- يكون بداية اسمه : **txtName** ماذا يعمل TextBox
- وكذلك بقية Controls لتسريع عملية البحث في الكود

لإغلاق Form الذي أنت عليه عن طريق **this.Close();**

الفرق بين () ShowDialog() vs Show()

- ❖ تستطيع الرجوع الى – الرئيسية – وإن لم تنتهي من Form1 **Show()**
- ❖ لا تستطيع الرجوع الى – الرئيسية – إلا بعد الانتهاء من Form1 **ShowDialog()**

```
1 reference
private void btnShowPart1_Click(object sender, EventArgs e)
{
    Form form1 = new Form1();
    form1.Show();
}

1 reference
private void btnShowFormAsDialog_Click(object sender, EventArgs e)
{
    Form form1 = new Form1();
    form1.ShowDialog();
}
```

تستخدم لعدة أشياء مثل MessageBox

- رسالة الى المستخدم تنبه على حدوث شيء
- استقبال مدخلات Input من المستخدم
- رسالة تأكيد للمستخدم – Ok & Cancel – .. وغيرها

لإظهار صندوق الرسالة نستخدم MessageBox.Show(Parameters)

لكل Parameter له عمل معين

١. رسالة الى المستخدم Message Parameter (text) .
٢. عنوان الرسالة Title Parameter (caption) .
٣. زر الإلغاء Cancel Parameter (buttons) .

```
I reference
private void button3_Click(object sender, EventArgs e)
{
    if( MessageBox.Show("Are you Sure ?",
    "Confirm !",MessageBoxButtons.OKCancel) == DialogResult.OK )
    {
        MessageBox.Show("User Pressed Ok");
    }
}
```

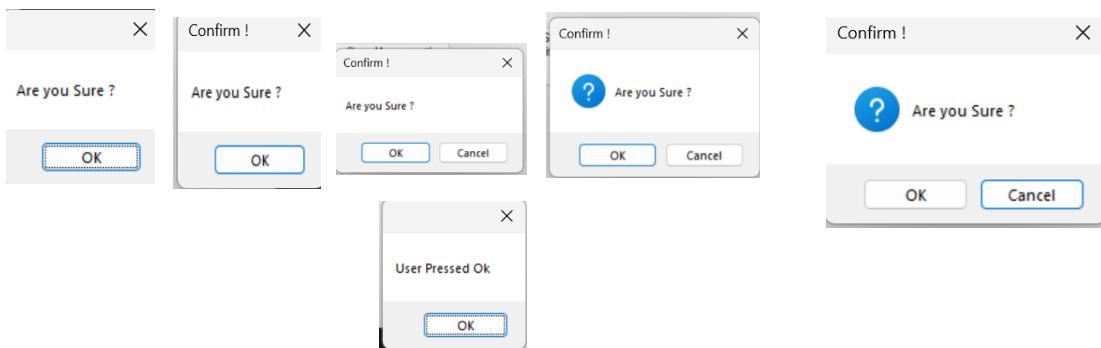
MessageBoxButtons.OKCancel .a

Tow Parameter فتظهر ok .b

للتتأكد بأن المستخدم اختار OK .c

الرابع : إضافة أيقونة بجانب الرسالة Parameter (icon) .d

الخامس : لجعل Tap يؤشر على Cancel Parameter (defaultButton) .e



❖ ومن أدواتها Common Controls ❖

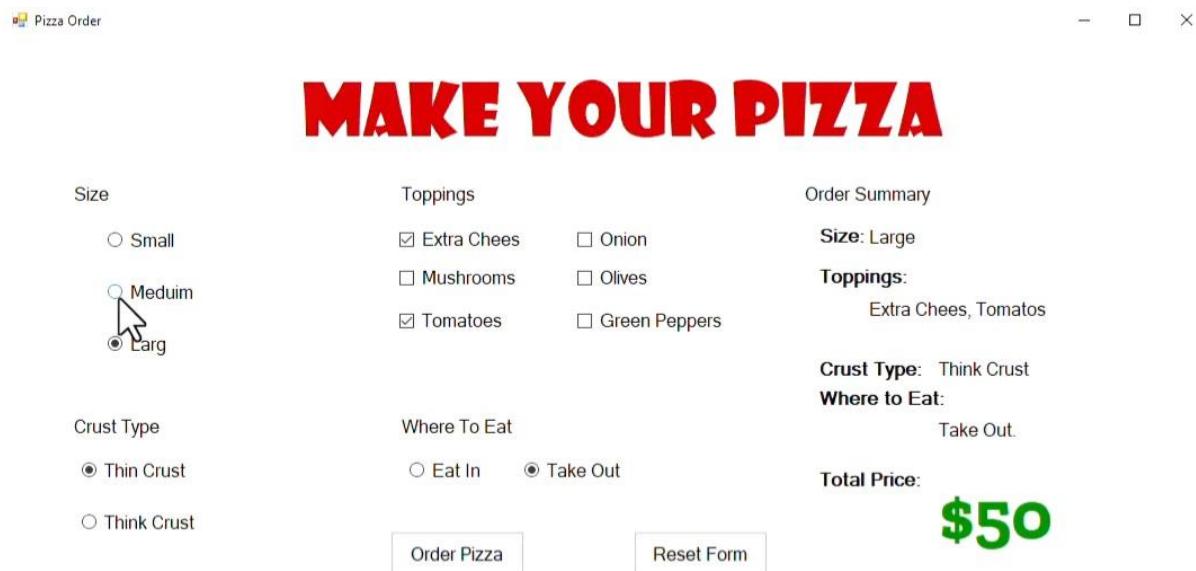
- لاختيار عدة خيارات – **Check box Control**
- للوصول الى قيمة **Check box** – اسمه ثم . ثم (True & False) Checked
- اختيار واحد من عدة اختيارات – **Radio Button**
- إضافة مجموعة من **Controls** تحت مجموعة واحدة – **Group box**
- تستخدم للتنظيم ولفصل **Radio** عن مجموعة أخرى أو أي آخر **Controls**
- كل لها **Property** مشتركة – تم شرح بعضها سابقاً
- ويوجد في كل **Property** ميزة **Controls** اسمها **Tag**
- هو متغير تستطيع تخزين فيه معلومات مهمة وهي **String** – تستطيع تحويلها الى أي نوع آخر – مثل ID

إذا أضفت مجموعة أخرى من **RadioButton** لابد من إضافتها داخل **Group box** لإظهار نتيجة على الشاشة يحول الى (.ToString())

قواعد في تسمية **Control** – الاسم الذي يتعامل معه في الكود (Name)



- يكون بداية اسمه : **btnName** ماذا يعمل **Button**
- يكون بداية اسمه : **txtName** ماذا يعمل **TextBox**
- يكون بداية اسمه : **chkName** ماذا يعمل **CheckBox**
- يكون بداية اسمه : **rbName** ماذا يعمل **RadioButton**
- يكون بداية اسمه : **gbName** ماذا يعمل **GroupBox**
- يكون بداية اسمه : **lblName** ماذا يعمل **Label**
- وكذلك بقية **Controls** لتسرير عملية البحث في الكود



رابط المشروع ما فإن أول خطوة هي تجزيء المشاكل الى مشاكل أصغر (فرق تسد)
<https://cdn.fs.teachablecdn.com/5sqLl00cRKq4VmDJCIE0> :

- ❖ يكون كل **GroupBox** مشكلة – لها متطلبات – مثل **Toppings** Function لتحويل السعر من نص الى رقم تم تخزينه في Tag
- ❖ لجمع أسعار الإضافات المختارة CalculateToppingsPrice() Function
- ❖ إرجاع ما تم اختياره في ملخص الفاتورة في نص واحد UpdateToppings() Function
- ❖ تحديث الملخص مع كل اختيار + (مجموع الفاتورة) - () UpdateTotalPrice()
- ❖ وكل **GroupBox** له نفس الخطوات السابقة تقريبا
- ❖ عند الضغط على Order Pizza يتم عمل Enabled Order Summary إلا الملخص GroupBox
- ❖ وأيضاً لـ Order Pizza : Button
- ❖ وإظهار رسالة تأكيد MessageBox لإنتهاء الطلب - نعم يعمل Enabled
- ❖ عند الضغط على Reset Form يتم إلغاء عمل Enabled
- ❖ ويتم إرجاع كل الاختيارات الى وضعها الطبيعي – مثل بداية البرنامج –
- ❖ Order Summary هو ملخص الفاتورة
- ❖ عند بداية عمل البرنامج يتم وضع الاختيارات المبدئية لبداية البرنامج (From1)

Lesson 40 : More about TextBox : Focus, MaxLength, Password, RightToLeft, Locked, Multiline , WordWrap

بعض الخصائص ل TextBox Properties

- ❖ **PasswordChar** : لتحويل النص الى رمز مثلا * أو أي شيء آخر - إخفاء النص وإبداله برمز
- ❖ **ReadOnly** : للقراءة فقط ، تختلف عن
- – تعديمه وعدم القدرة على اختياره **Enabled**
- ❖ **MaxLength** : تحديد عدد الحروف المسموح بكتابتها
- ❖ **TextAlign** : تحديد محاذاة النص سواء يمين ، يسار ، وسط وتختلف عن
- – تحديد اتجاه الكتابة **RightToLeft**
- ❖ **Multiline** : تتحكم في حجم TextBox + للكتابة في أكثر من سطر
- ❖ **WordWrap** : للنزول الى سطر جديد عند نهاية السطر - أو نهاية حجم TextBox

بعض الخصائص ل Controls Properties – المشتركة

- ❖ **RightToLeft** : تحديد اتجاه الكتابة
- إذا تم اختيار الوراثة فيرث من parent الذي يحتوي TextBox
- ❖ **Tag** : هو متغير تستطيع تخزين فيه معلومات مهمة وهي String – تستطيع تحويلها الى أي نوع آخر - مثل ID
- ❖ **Control** : تثبيت Control في مكانه وعدم تحريكه في Desing
- ❖ **Dock** : لتحديد كيفية وضع Control داخل الذي يحتويه
- None: لا يتم تثبيت العنصر في أي مكان.
- Top: يتم تثبيت العنصر في الجزء العلوي من الحاوية.
- Bottom: يتم تثبيت العنصر في الجزء السفلي من الحاوية.
- Left: يتم تثبيت العنصر على يسار الحاوية.
- Right: يتم تثبيت العنصر على يمين الحاوية.
- Fill: يملأ العنصر المساحة المتاحة في الحاوية.

❖ **ControlName.Focus();** : للذهاب الى Control الذي يتفاعل معه المستخدم

❖ ومن أدواتها **Common Controls** ❖
○ لإضافة صور الى البرنامج : **PictureBox** ○

❖ **بعض الخصائص ل TextBox Properties** ❖

- **Image** : لاختيار صورة من طريق معين
- **SizeMode** : للتحكم بعرض الصورة داخل المربع
- **AutoSize** : تغيير حجم الصورة لتناسب الذي يحتويها
- **CenterImage** : عرض الصورة من المنتصف
- **StretchImage** : ملء الصورة لكامل المربع الذي يحتويها
- **Zoom** : عرض الصورة بشكل كامل حتى لو صغر المربع الذي يحتويها

pbName يكون بداية اسمه : **PictureBox**

يوجد عدة طرق لإضافة صور الى البرنامج :

- ١ . **project resource file** : عند اختيار صورة معينة من الكمبيوتر ، يتم إضافة مجلد جديد داخل ملفات البرنامج يكون اسمه **Resource** وإضافة الصور بداخل المجلد
 - a. التعامل معها سهل
 - b. يتم استخدام اسم الصورة في الكود - سمها باسم معبر عن محتواها -
 - c. للوصول الى الصورة عبر الكود : اسم الملف المخزن فيه الصورة . اسم الصورة **Resources.Strong** ; مثال :
 - d. ومن عيوبها : أنها تكبر حجم البرنامج
- ٢ . **Local resource** : تحميل الصورة من الجهاز مثل (C & D) drive (مسار الصورة من الجهاز " @)
a. للوصول الى الصورة عبر الكود : " مسار الصورة من الجهاز " @ (Image.FromFile())
b. @ : لتنسيق النصوص لتجاوز استخدام " , \

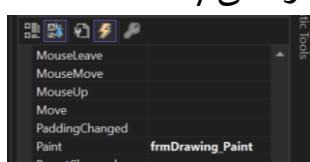
Lesson 42 & 43 : PictureBox Exercise & (Solution)

رابط المشروع : <https://cdn.fs.teachablecdn.com/wY4MUy4MQ8Kc58uEQE6c>

❖ Event \ Method هو موجود في كل Parameter object sender
❖ Event \ Method الذي استدعي Control هو sender ○
▪ لذا لابد من عمل Casting لتحديد نوع sender
• مثال ; ((RadioButton)sender).Tag.ToString()
○ تستطيع استخدامه مع أكثر من sender

Lesson 44 : Draw Line , ellipse , and rectangle .

❖ تستطيع الرسم في C# من خلال مكتبة اسمها : Color
❖ لتحديد لون الخط Color.FromArgb(255, 0, 0) - اللون الأسود –
❖ تحكم سماكة الشكل – القلم – عن طريق Width :
❖ تحدد له لون الخط وسماكته وشكل بداية الخط وتسمى x & y
❖ وكذلك النهاية له شيئاً يسمى x & y
❖ للرسم تذهب الى Events من قائمة ⚡ ثم
❖ اختيار Paint – ضغطتين فيذهب الى Event \ Method
❖ لرسم الأشكال على الشاشة : e.Graphics.DrawLine(pen, 100, 100, 100, 200);

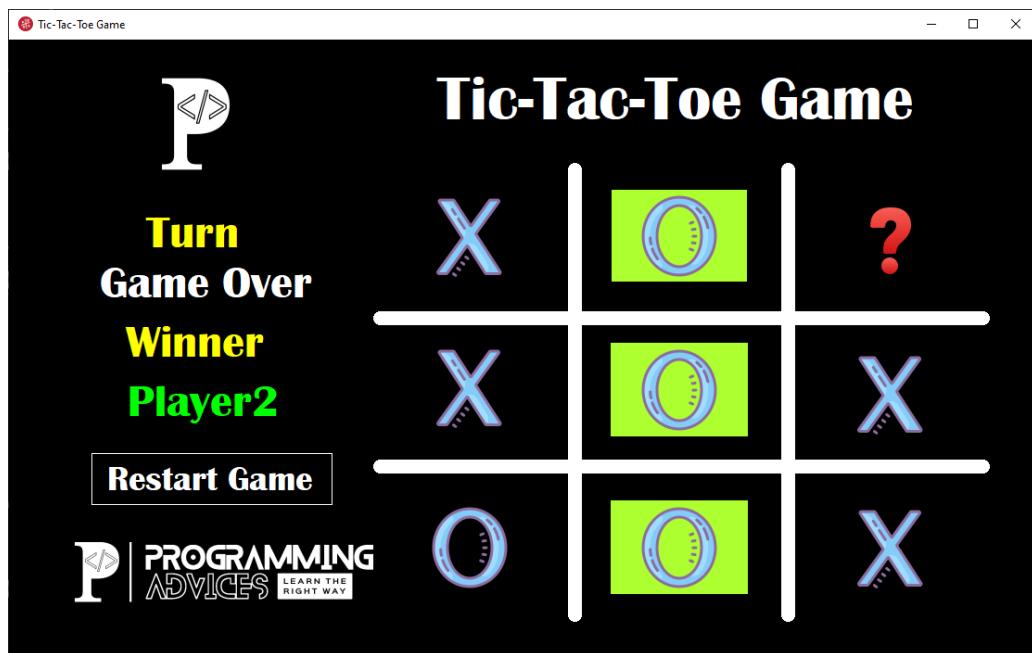


```
private void frmDrawing_Paint(object sender, PaintEventArgs e)
{
    Color Black = Color.FromArgb(255, 0, 0, 0);
    Pen pen = new Pen(Black);
    pen.Width = 10;
    // pen.DashStyle=System.Drawing.Drawing2D.DashStyle.Dash;
    pen.StartCap = System.Drawing.Drawing2D.LineCap.Round;
    pen.EndCap = System.Drawing.Drawing2D.LineCap.Round;

    e.Graphics.DrawLine(pen, 100, 100, 100, 200);
    e.Graphics.DrawRectangle(pen, 200, 200, 300, 300);
    e.Graphics.DrawEllipse(pen, 200, 50, 100, 120);

}
```

Lesson 45 & 46 : Tic-Tac-Toe Game (Requirement / Solution)



رابط المشروع : <https://cdn.fs.teachablecdn.com/tDjUjvnQAYKLeFM5mAUH>

Lesson 47 : Tic-Tac-Toe Game Solution One Event

إذا كانت جميع Controls أو بعضها لها عمل موحد : تستطيع اختصار الكود ب واحد Event

مثال button 1 - 9 تستدعي Function واحد ChangeImage(button 1-9)

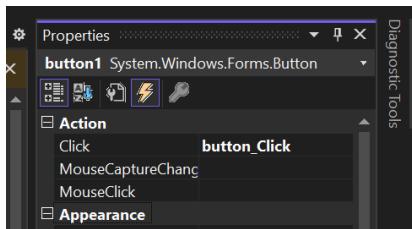
❖ عمل Event جديدة وحذف جميع Event الأخرى

```
private void button_Click(object sender, EventArgs e)
{
    ChangeImage( (Button) sender);
}
```

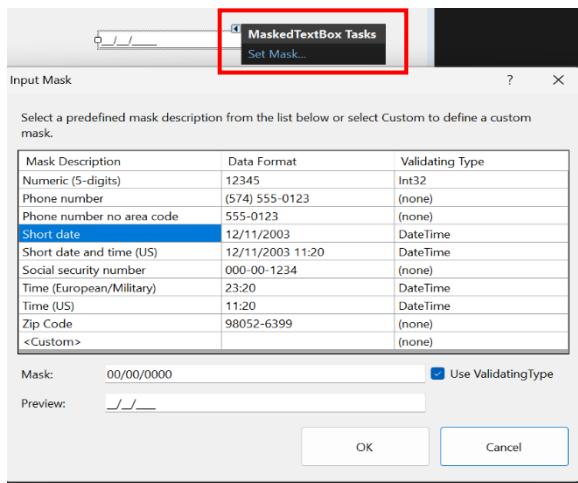
❖ ثم نسخ اسم Event الجديدة ولصقها في Click

○ تذهب الى Events ثم Click

❖ sender تخزن أي button الذي استدعاه
و يتم تحويل sender لنوع Control الذي استدعاه



Lesson 48 : MaskedTextBox



Common Controls ❀ ومن أدواتها

- **MaskedTextBox** : إدخال بيانات بتنسيق معين للحصول على تنسيق
- ولابد من الالتزام بإدخال البيانات المطلوبة مثل التاريخ أرقام ...
- يوجد **Property BeepOnError** تعطيك تنبيه - صوت - إذا كان المدخل خطأ اسمها
- والتعامل معها في الكود مثل أي **Control** آخر
- **MaskedTextBox** تستخدم في برامج المحاسبة وغيرها - الإدخال - (قد لا تظهر بعض Mask التي في الصورة والحل تحويل لغة الجهاز نفسه من الإعدادات إلى English)

يوجد اسمها **Mask** لتنسيق المدخلات وهي إما رقم أو حرف أو رمز وكل واحد عمله وهذا التنسيق يكون ثابتاً على Mask ولو كان في وقت تشغيل البرنامج ، لا يمكن للمستخدم نقلها أو حذفها

- 0 : إدخال إجباري لرقم بين 0 و 9
- 9 : إدخال إجباري لرقم بين 0 و 9 أو مسافة
- # : إدخال إجباري لرقم بين 0 و 9 أو مسافة
- A : إدخال إجباري لحرف كبير - أو صغير أو رقم أو مسافة -
- a : إدخال إجباري لحرف صغير - أو كبير أو رقم أو مسافة -
- L : إدخال إجباري لحرف كبير أو صغير - أو رقم أو مسافة -
- ? : إدخال إجباري لحرف كبير - أو صغير أو مسافة -
- & : إدخال إجباري لحرف كبير - أو صغير أو رقم أو مسافة - أو رمز
- C : إدخال إجباري لحرف كبير - أو صغير أو رقم أو مسافة - أو رمز ثابت للفصل بين الوقت
- / رمز ثابت للفصل بين التاريخ
- \\$ رمز ثابت للعملة
- < تحويل الحروف التالية إلى صغيرة
- > تحويل الحروف التالية إلى كبيرة
- | تعطيل التحويل من الحروف الصغيرة إلى الكبيرة أو العكس

RadioButton قد يكون بديلاً لـ **Drop down list** أو **ComboBox**

في **Properties** منها **ComboBox** يوجد

- إضافة عناصر على **Item**
- للتحكم بالقائمة **Drop down Style**
 - ظهور كل العناصر **Simple**
 - تبحث أو تختار من القائمة – **Drop down**
 - عدم الكتابة في **ComboBox** – **Drop down List**
 - تختار فقط – **Sorted** لترتيب العناصر – أبجدي

عند اختيار عنصر معين من القائمة – يتم مناداة **Function** مع كل اختيار وإرجاع العنصر مع حدث **Event**

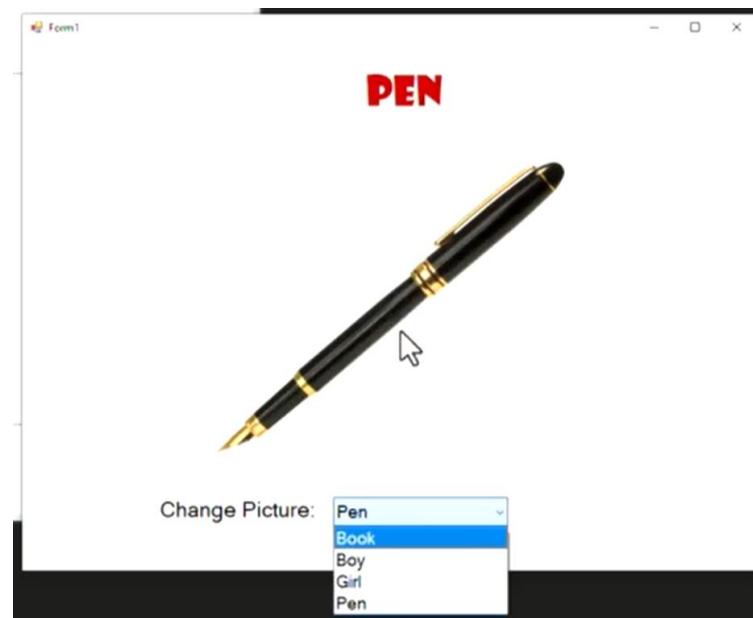
```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    MessageBox.Show(comboBox1.Text);
}
```

لجعل خيار معين هو الخيار المبدئي عند تشغيل البرنامج

```
private void frmComboBox_Load(object sender, EventArgs e)
{
    comboBox1.SelectedIndex = 1;
}
```

cmbName يكون بداية اسمه : **ComboBox**

Lesson 50 : Combo Box Exercise / Solution



رابط المشروع : <https://cdn.fs.teachablecdn.com/mWbgnqvSH62gmMV2FtK4>

```
private void cmbPictures_SelectedIndexChanged(object sender, EventArgs e)
{
    switch (cmbPictures.SelectedItem.ToString().ToLower())
    {
        case "boy":
            pictureBox1.Image = Resources.Boy;
            lblTitle.Text = "Boy";
            break;
        case "girl":
            pictureBox1.Image = Resources.Girl;
            lblTitle.Text = "Girl";
            break;
        case "book":
            pictureBox1.Image = Resources.Book;
            lblTitle.Text = "Book";
            break;
        case "pen":
            pictureBox1.Image = Resources.Pen;
            lblTitle.Text = "Pen";
            break;
    }
}
```

Lesson 51 : LinkLabel

للذهاب الى موقع معين عبر رابط **LinkLabel**

في **Properties** يوجد منها **LinkLabel**

- التحكم بمظهر الخط أو عدمه **Link Behavior**
- لاحتواء النص بشكل تلقائي **Auto Size**

عند الضغط على **LinkLabel** يذهب الى الموقع عبر المتصفح الافتراضي للجهاز

```
private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
{
    linkLabel1.LinkVisited = true;
    System.Diagnostics.Process.Start("http://www.ProgrammingAdvices.com");
}
```

Lesson 52 : CheckedListBox

قد يكون بديلاً **CheckBox** **CheckedListBox**

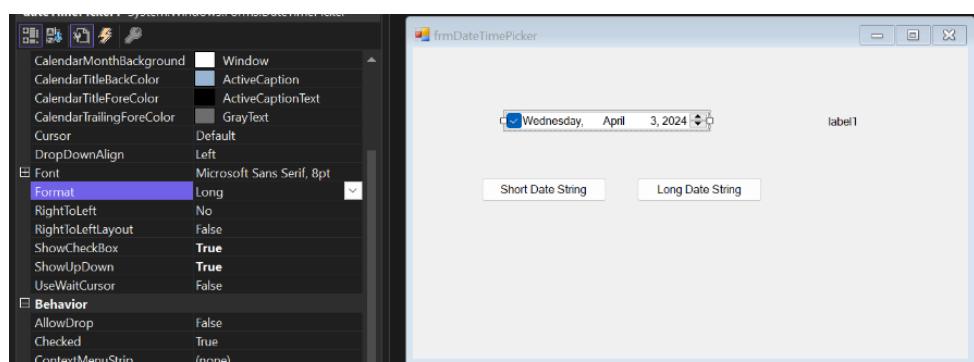
تستطيع إضافة **Item** إما عن طريق **Property** أو عن طريق الكود ("")

لإرجاع اسم لعنصر معين في القائمة `checkedListBox1.CheckedItems[i]`

لاختيار أو عدمه `checkedListBox1.SetItemChecked(i, true)`

لحذف عنصر محدد `checkedListBox1.Items.RemoveAt(2);`

Lesson 53 : DateTimePicker



Date Time Picker

تحديد يوم من التقويم

من في Property
DateTimePicker

- لاختيار التنسيق التقويم - طويل اسم اليوم والشهر / قصير أرقام /
- تنسيق خاص بك - انت الذي تحدده -
- True - لا تستطيع اختيار يوم إلا بعد عمل على التقويم
- True - أخفاء مربع التقويم لاختيار يوم ، الاختيار عن طريق الأسهم ⇩
- **MaxDate / MinDate** : لتحديد تاريخ يقع بينهما

لطباعة التاريخ طويل أو صغير

```
MessageBox.Show(dateTimePicker1.Value.ToShortDateString());
MessageBox.Show(dateTimePicker1.Value.ToString());
```

جمع التواريف في نص واحد مع النزول لسطر جديد لكل تاريخ

```
label1.Text = dateTimePicker1.Value.ToString() + Environment.NewLine;
label1.Text += dateTimePicker1.Value.ToString("dd-MMM-yyyy") +
Environment.NewLine;
label1.Text += dateTimePicker1.Value.ToString("dddd-MM-yyyy") +
Environment.NewLine;
label1.Text += dateTimePicker1.Value.ToString("MM-dd-yyyy") +
Environment.NewLine;
label1.Text += dateTimePicker1.Value.ToString("dd-MM-yy") +
Environment.NewLine;
label1.Text += dateTimePicker1.Value.ToString("ddd, dd-MM-yyyy") +
Environment.NewLine;
```

للنزول الى سطر جديد `Environment.NewLine`

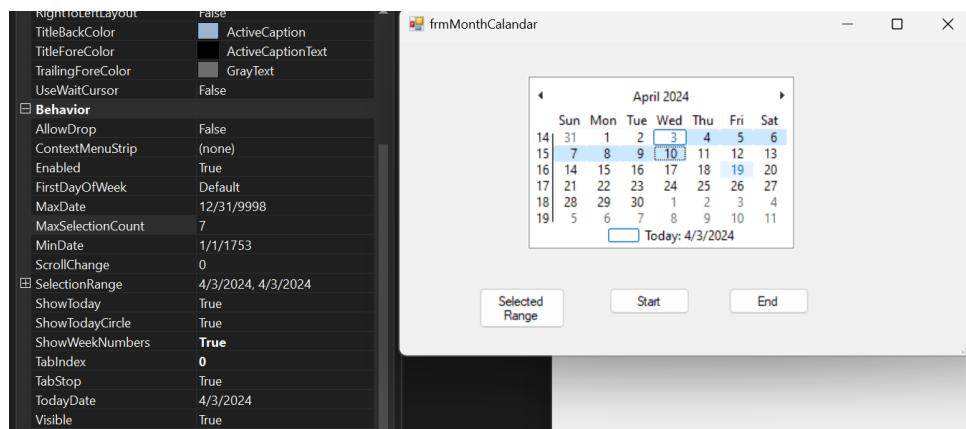
(أرقام من خانتين أو من خانة واحدة إذا كانت أقل من 10) `d = 0 \ M = 00 \ MM = dd- MMM`

`Fri – Apr = ddd- MMM / Friday – April = dddd- MMMM` = الاسم بالكامل

Format مثل DateTimePicker مع Property تشارك في بعض MonthCalandar

من MonthCalandar في Property

- كم عدد الأيام المتتالية المسموح بها لاختيارها مثل الحجوزات •
- إظهار تاريخ اليوم True : Show To Day •
- ترقيم الأسبوع - رقم الأسبوع من بداية السنة - Show Week Number •

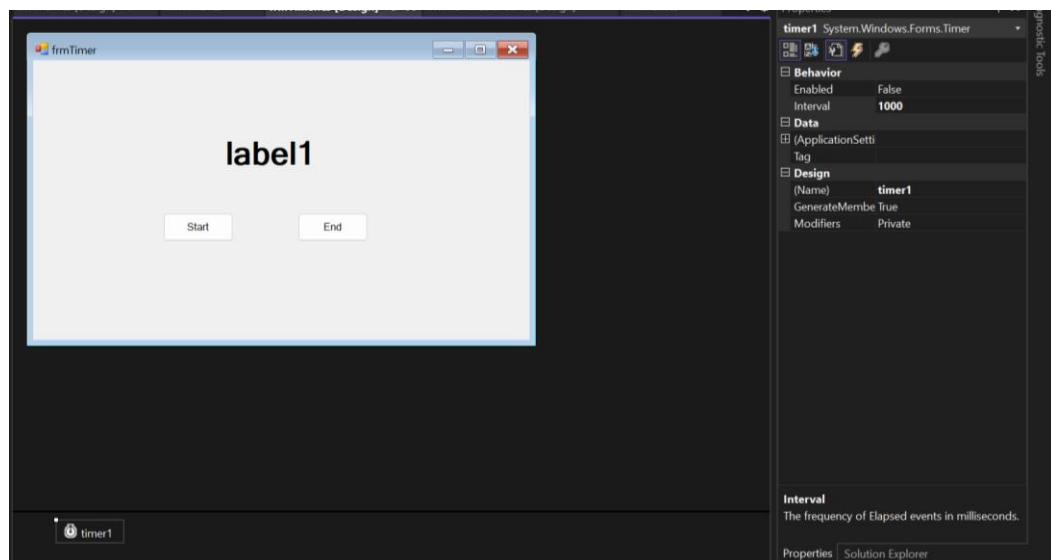


Lesson 55 : Timer

عبارة عن مؤقت يتم تشغيله إذا كان Enabled على True – يتم الحساب على حسب 1000 = Interval يعني ثانية واحدة –

Interval يسمى Event Function – Timer في him يدعى Tick

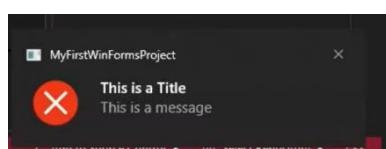
يظهر Timer Design أسلف



Lesson 56 : NotifyIcon

Balloon هو إشعار يظهر في قائمة الإشعارات للجهاز ويسمى :

يظهر NotifyIcon مثل Timer Design أسلف



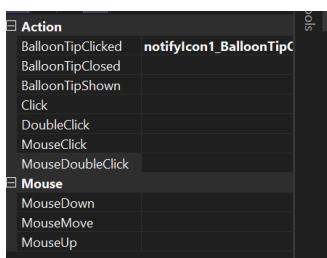
مدة ظهور NotifyIcon بين 10 و 30 ثانية فقط

يتكون من NotifyIcon

ايقونة للبرنامج / و ايقونة للرسالة مثل ❓ وغيرها / العنوان / والرسالة

لكل 🔍 Event عمل خاص مثل عند الضغط على الإشعار أو إغلاقه وغيرها

لابد من تفعيل الإشعارات للجهاز لظهور الإشعار



Control تضييف فيها مجموعة من الصور والتحكم بحجمها و تستطيع ربطها مع أكثر من ImageList بعض Property ل

- Index : لإضافة الصور ، تكون مرقمة من 0
- ImageSize : عرض حجم الصورة ، الطول ، العرض

Node Child هي مثل الشجرة ، ويسمى كل عنصر Node وكل فرع منها ب TreeView و تستطيع جعل كل أعضاء Checked على Tree View

- أو الوصول إليها عن طريق الكود - بعض Property ل

- Nodes : لإضافة أعضاء Node
- ImageList : إضافة صور من Nodes لكل ImageList
- Property Image عن طريق Node معينة ل Tree View ل اختيار Image مع كل حدث Tree View Event
- SelectedImageIndex Properties وغيرها من SelectedImageIndex
- CheckBoxes : لوضع اختيار المربع بجانب Node

بعض TreeView Event ل

MouseDoubleClick : الحدث هو عند الضغط على Node مرتين - ينفذ الأمر -
AfterCheck : عند اختيار Node أو عدمه

```
private void CheckTreeNode(TreeNode node, Boolean isChecked)
{
    foreach (TreeNode item in node.Nodes)
    {
        item.Checked = isChecked;

        if (item.Nodes.Count > 0)
        {
            this.CheckTreeNode(item, isChecked);
        }
    }
}

private void treeView1_AfterCheck(object sender, TreeViewEventArgs e)
{
    CheckTreeNode(e.Node, e.Node.Checked);
}
```

شريط يظهر نسبة اكتمال الإنجاز مثل شريط التحميل : **ProgressBar**

بعض **ProgressBar** ل **Property**

- البداية : **Minimum**
- انتهاء الشريط : **Maximum**
- إضافة قيمة معينة لتعبئة الشريط : **Value**

```
private void button1_Click(object sender, EventArgs e)
{
    progressBar1.Value = 0;
    progressBar1.Maximum = 100;
    progressBar1.Minimum = 0;

    for (int i=0;i<=10;i++)
    {
        if (progressBar1.Value < progressBar1.Maximum)
        {
            // لتبطئه سرعة البرنامج مثل 500 = نصف ثانية
            // لابد من استدعاء مكتبة System.Threading;
            Thread.Sleep(500);
            progressBar1.Value += 10;

            label1.Text = (((float)progressBar1.Value /
progressBar1.Maximum) * 100) + "%";

            // Refresh()
            // لأنه قد تم تبطئه البرنامج + أنه يعمل تحديث للشاشة
            progressBar1.Refresh();
            label1.Refresh();
        }
        else
        {
            button1.Enabled = false;
        }
    }
}
```

هي طريقة عرض العناصر - ملفات - على الشاشة مثل تكبير أو تصغير Icon وغيرها

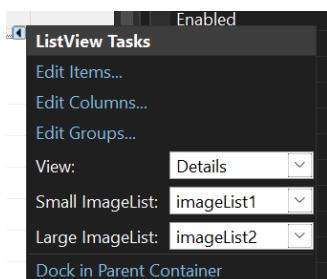
يمكن عرض العناصر باستخدام أربعة طرق مختلفة :

- عرض العناصر باستخدام أيقونات كبيرة.
- عرض العناصر باستخدام أيقونات صغيرة.
- عرض العناصر في قائمة نصية.
- عرض العناصر في جدول مع أعمدة وصفية.

تستطيع إضافة أكثر من Sub Item ل Item الرئيسي

يوجد في Form Property Accept Button مهملته عند الضغط على Enter يتم تنفيذ الكود

الأفضل في تطبيقات Desktop أنك تشتعل على التطبيق من غير ماوس



لضبط عرض العناصر بشكل تلقائي – View

الفرق بينهما في حجم أبعاد الصورة Image

```
private void btnAdd_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(txtID.Text) ||
        string.IsNullOrEmpty(txtName.Text))
        return;

    ListViewItem item = new ListViewItem(txtID.Text.Trim());
    if (rbMale.Checked)
        item.ImageIndex = 1;
    else
        item.ImageIndex = 0;

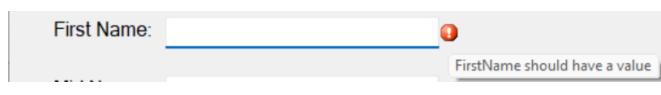
    item.SubItems.Add(txtName.Text.Trim());
    listView1.Items.Add(item);

    txtID.Clear();
    txtName.Clear();
    txtID.Focus();
}
```

Lesson 60 : ErrorProvider

Validation للتحقق من المدخل وهو افضل من MessageBox ، فيظهر الخطأ مباشرةً إذا تم تجاوزه وعدم إدخال شيء و تستطيع استخدام ErrorProvider الواحد لأكثر من text Box بعض ErrorProvider ل Property

- لإظهار معدل الوميض Blink Rate



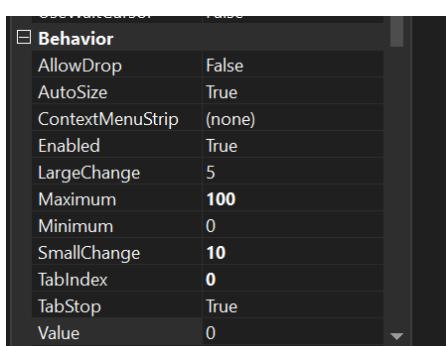
يوجد في textBox Evant اسمه

- يتم مناداته عند الخروج من Control الى آخر Control Validating

Lesson 61 : TrackBar



هو مثل رفع الصوت أو خفضه TrackBar



بعض TrackBar ل Property

- البداية : Minimum
- انتهاء الشرط : Maximum
- إضافة قيمة معينة لتعبئة الشرط : Value
- لإضافة القيمة عبر أو : Small Change
- لإضافة القيمة عبر أو : Large Change

Lesson 62 : NumericUpDown

أو كتابة الرقم في الحقل  أو  لزيادة قيمة أو نقصها مع ضغطة **NumericUpDown**



بعض **NumericUpDown** Property لـ

- البداية : **Minimum**
- النهاية : **Maximum**
- إضافة قيمة معينة لزيادة قيمة أو نقصها : **Increment**

Lesson 63 : TabControl

ترتيب البيانات بطريقة منظمة – كل صفحة مثل Form تستطيع إضافة أي شيء –

– وهي توفر وقت كثير في Desdign

عنصر للتحكم في واجهة forms وجعلها متعددة الصفحات داخل نفس الفورم (النافذة)

بعض **TabControl** Property لـ



- لإظهار كل الصفحات مع **tabsPages**
- عرض محاذاة لقوائم الصفحات : **Alignment**

Lesson 64 : GroupBox vs Panel



في **GroupBox** لا تستطيع إخفاء **Title**

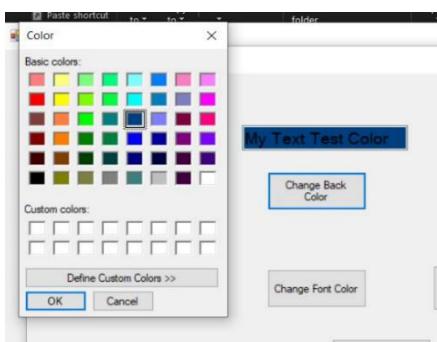
أما **Panel** فهو مثل **GroupBox** لكن بدون **Title** و **border**



وفيه شريط تمرير **scroll par**

يأخذ **Drag drub** – ستدرس لاحقا –

Lesson 65 : ColorDialog



هو صندوق ألوان لتغيير لون الخلفية أو النص **ColorDialog**

سواء TextBox Or Form

يسمح لك باختيار لون في **ColorDialog** run time

```
private void button1_Click(object sender, EventArgs e)
{
    //Check the color selection from colorDialog control
    if (colorDialog1.ShowDialog() == DialogResult.OK)
    {
        //colorDialog1.Color الون المختار
        textBox1.BackColor = colorDialog1.Color;
        // Text box backcolor سيقوم بتغيير لون في
    }
}
```

Lesson 66 : FontDialog

هو لتغيير لون الخلفية أو النص أو لتغيير حجم ولون وشكل الخط **FontDialog**
يظهر لك **Front** على حسب النظام

```
private void button3_Click(object sender, EventArgs e)
{
    fontDialog1.ShowApply = true;
    fontDialog1.ShowColor = true;
    fontDialog1.ShowEffects = true;
    fontDialog1.ShowHelp = true;

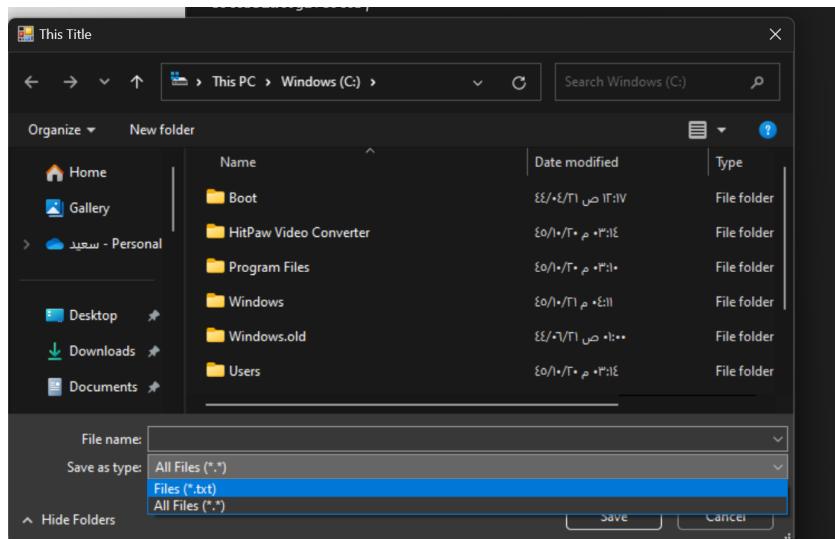
    // لكيلا يحدث خطأ fontDialog1.Font وتخزينه في textBox1
    fontDialog1.Font = textBox1.Font;

    if (fontDialog1.ShowDialog() == DialogResult.OK)
    {
        textBox1.Font = fontDialog1.Font;
        textBox1.ForeColor = fontDialog1.Color;
    }
}

private void fontDialog1_Apply(object sender, EventArgs e)
{
    textBox1.Font = fontDialog1.Font;
    textBox1.ForeColor = fontDialog1.Color;
}
```

Lesson 67 : Save File Dialog

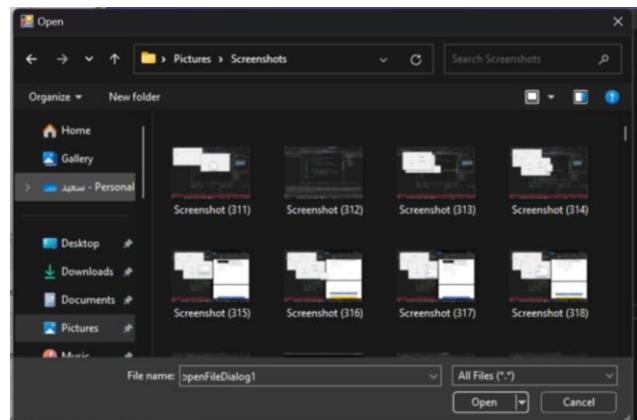
يرجع لك **Path** المسار التي تم فيه حفظ الملف مثل الورد عند حفظ الملف
جديد - يفتح الملفات لتحديد موقع الحفظ -



```
private void button4_Click(object sender, EventArgs e)
{
    // يجعل البداية عند فتح الملفات عند هذا المسار
    saveFileDialog1.InitialDirectory = @"C:\";
    // تغيير العنوان
    saveFileDialog1.Title = "This Title / Save";
    // إضافة امتداد للملف بشكل افتراضي
    saveFileDialog1.DefaultExt = "txt";
    // لقائمة الملفات أو اظهار امتداد الملفات معينة مثل ...
    saveFileDialog1.Filter = "Files (*.txt) | *.txt | All Files (*.*) | *.*";
    // تحديد فلتر معين بشكل تلقائي مثل (*.*)
    saveFileDialog1.FilterIndex = 2;
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        MessageBox.Show(saveFileDialog1.FileName);
    }
}
```

Lesson 68 : Open File Dialog

يرجع لك **Path** مسار الملف الذي تريد فتحه **FileOpenDialog**



```
private void button5_Click(object sender, EventArgs e)
{
    // يجعل البداية عند فتح الملفات عند هذا المسار
    openFileDialog1.InitialDirectory = @"C:\\";

    // لتغيير عنوان
    openFileDialog1.Title = "This Title / Open ";

    // لاصافة امتداد للملف بشكل افتراضي
    openFileDialog1.DefaultExt = "txt";

    // لفلترة الملفات أو اظهار امتداد الملفات معينة مثل ...
    openFileDialog1.Filter = " Files (*.txt) | *.txt | All Files (*.*) | *.*";

    // تحديد فلتر معين بشكل تلقائي مثل (*.*)
    openFileDialog1.FilterIndex = 2;

    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        MessageBox.Show(openFileDialog1.FileName);
    }
}

private void button6_Click(object sender, EventArgs e)
{
    // لاختيار أكثر من ملف
    openFileDialog1.Multiselect = true;

    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        foreach ( string file in openFileDialog1.FileNames )
        {
            MessageBox.Show(file);
        }
    }
}
```

Lesson 69 : Folder Browser Dialog

يرجع لك **Path** مسار المجلد فقط وليس الملفات الذي تريد فتحه

```
private void button7_Click(object sender, EventArgs e)
{
    // لإظهار زر إضافة مجلد جديد
    folderBrowserDialog1.ShowNewFolderButton = true;

    if(folderBrowserDialog1.ShowDialog() == DialogResult.OK)
    {
        MessageBox.Show(folderBrowserDialog1.SelectedPath);
    }
}
```



Lesson 70 : MDI Container

اختصار **MDI** **Multiple Documents Interface**

- يجعل **Form2** يخرج من **Form1** لا يدخل **Form1** - مثال عند تصغيره يتم تصغير الكل

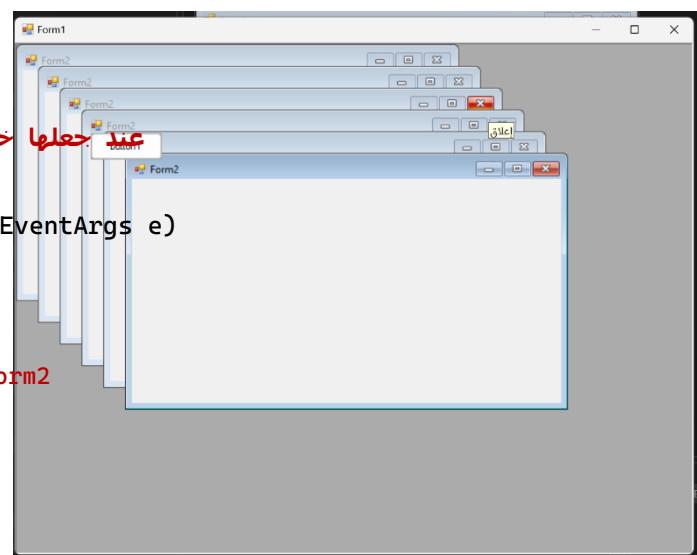
- فيوجد **Form** لـ **Property** اسمها **IsMdiContainer** - لتضيف **Form** آخر بداخلها

عادة يتم إضافة **Button** **Form MDI** في **Menue** وليس **Form**

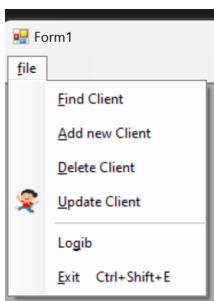
```
// عند جعلها خارج button1_Click يتم اظهارها مرة واحدة فقط
private Form form2 = new Form2();

private void button1_Click(object sender, EventArgs e)
{
    // مع كل استدعاء يتم انشاء Form2 جديد
    //Form form2 = new Form2();

    // Form1 (this) هو Object تبع هذا Form2
    form2.MdiParent = this;
    form2.Show();
}
```



Lesson 71 : ToolStrip



ـ مثل Menu لـ Form في Visual studio إضافة قائمة Menu إلى ToolStrip

ToolStrip

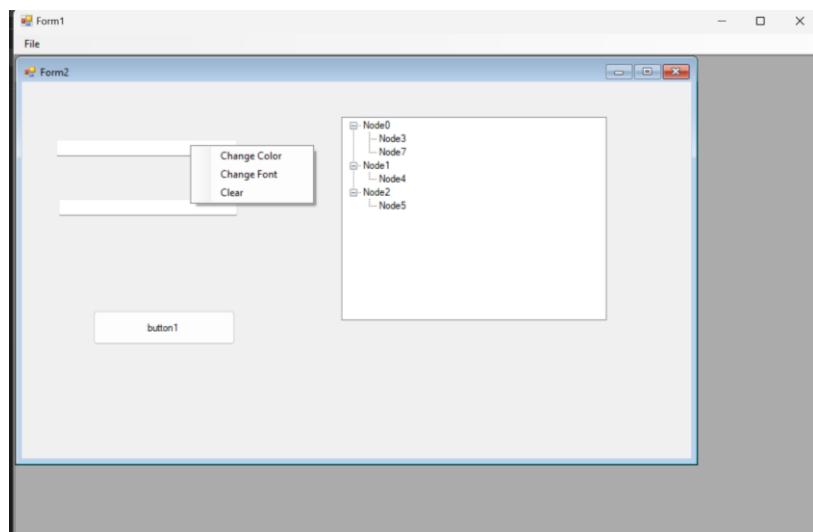
- ❖ لإضافة صورة بجانب Menu – بزر الفأرة الأيمن ثم اختيار Set Image
- ❖ تستطيع إضافة Menu فرعية

- ❖ لإضافة اختصار بالكيبورد مثل Ctrl + Shift + E للخروج من البرنامج – عن طريق Short Cut Keys
- ❖ عند الضغط على Alt في Visual studio يتم الذهاب إلى قائمة Menu مع إظهار خط تحت كل حرف من Menu ثم اختيار أي حرف منها للذهاب إلى Menu الخاص بها ، كذلك تستطيع عملها في برنامجك بمعنى آخر فتح Menu عن طريق الكيبورد
- **طريقة عملها وضع علامة & قبل الحرف الذي تريد جعله اختصارا**

وغيرها من Property

Lesson 72 : Context Menu

ـ TextBox : تظهر قائمة Menu حسب الموضع بزر الفأرة الأيمن – مثلاً في TextBox



ولابد من ممارسة كل الدروس التي تم أخذها

الفهرس

صفحة	العنوان
1	<u>#Lesson 1 : About this Course & Installing C#</u>
2	<u>#Lesson 2 : Library vs Framework vs Platform</u>
3	<u>#Lesson 3 : What is .NET ? .NET Core vs .NET Framework ? What can we do with .NET ?</u>
6	<u>#Lesson 4 : What to Learn .NET Framework or .NET Core</u>
7	<u>#Lesson 5 : Compilation in .NET</u>
10	<u>#Lesson 6 : .NET Framework Architecture</u>
11	<u>#Lesson 7 : What is CLR</u>
12	<u>Lesson 8 : CLR Main Components</u>
12	<u>Lesson 9 : 1- Common Type System (CTS)</u>
13	<u>Lesson 10 : 2- Common Language Specification (CLS)</u>
13	<u>Lesson 11 : 3- Garbage Collector (GC)</u>
14	<u>Lesson 12 : 4- Just In Time Compilation (JIT)</u>
16	<u>Lesson 13 : 5 and 6: Assemblies & Metadata(Manifest)</u>
18	<u>Lesson 14 : CLR Functions</u>
19	<u>Lesson 15 : Managed vs Unmanaged Code</u>
20	<u>Lesson 16 : CLR Structure</u>
22	<u>Lesson 17 : .NET Framework Class Library (FCL)</u>

23	<u>Lesson 18 : What is C# ? and Why to Learn It</u>
25	<u>Lesson 19 : Syntax</u>
26	<u>Lesson 20 : WriteLine</u>
26	<u>Lesson 21 : Write</u>
27	<u>Lesson 22 : Formatted String</u>
27	<u>Lesson 23 : Escape Characters (\)</u>
28	<u>Lesson 24 : Single Line/Multiple Lines Comments (/ /, /* */)</u>
28	<u>Lesson 25 : Variables</u>
29	<u>Lesson 26 : Rules for Naming Variables in C#</u>
30	<u>Lesson 27 : Implicitly Typed Variables</u>
30	<u>Lesson 28 : Datatypes</u>
31	<u>Lesson 29 : Predefined Datatypes</u>
32	<u>Lesson 30 : Numbers Datatypes</u>
35	<u>Lesson : Default Values</u>
36	<u>Lesson : Enum</u>
37	<u>Lesson : Nullable Types</u>
38	<u>Lesson : Anonymous Type</u>
39	<u>Lesson : Structures</u>
40	<u>Lesson : Dynamic Type</u>
40	<u>Lesson : Set Date & Time</u>
41	<u>Lesson : Get Current Datetime</u>
41	<u>Lesson : Ticks</u>

42	<u>Lesson : DateTime Static Fields</u>
42	<u>Lesson : Time Span</u>
43	<u>Lesson : Subtraction of two dates results in TimeSpan</u>
43	<u>Lesson : Operators</u>
44	<u>Lesson : Convert String to DateTime</u>
45	<u>Lesson : Quick Overview</u>
45	<u>Lesson : String Interpolation</u>
46	<u>Lesson : Casting Types : (Implicit Casting / Explicit Casting)</u>
47	<u>Lesson : Type Conversion Methods</u>
47	<u>Lesson : Casting Enums</u>
48	<u>Lesson : ReadLine</u>
48	<u>Lesson : User Input and Numbers</u>
49	<u>Lesson : (Assignment / Arithmetic / Relational / Logical) Operators</u>
51	<u>Lesson : Unary Operators / Ternary Operator</u>
52	<u>Lesson : Bitwise and Bit Shift Operators</u>
53	<u>Lesson : if, if...else, if...else if and Nested if Statement</u>
53	<u>Lesson : switch Statement / ternary (?) Operator / for loop</u>
54	<u>Lesson : while loop / do while loop / (break / continue) Statement</u>
55	<u>Lesson : Arrays Are Bound / Array Declaration</u>

56	<u>Lesson : Array initialization / Access Array Elements</u>
57	<u>Lesson : Two-Dimensional Array</u>
57	<u>Lesson : for each loop</u>
58	<u>Lesson : Array Operations using System.Linq</u>
59	<u>Lesson : C# Math</u>
59	<u>Lesson : C# Methods</u>
60	<u>Lesson : Method Parameters / Default Parameter Value</u>
61	<u>Lesson : Return Values / Named Arguments</u>
62	<u>Lesson : Method Overloading</u>
63	<u>Lesson : Exceptions / Random Function In C#</u>
64	<u>Syntax C# vs C++</u>
66	<u>Lesson 31 : Web vs Desktop Applications</u>
70	<u>Lesson 32 : Introduction About Controls – Part1</u>
71	<u>Lesson 33 : Part 2 - First Windows Forms Application</u>
72	<u>Lesson 34 : Introduction About Controls – Part3</u>
73	<u>Lesson 35 : Part 4 : Some formatting and Alignments</u>
74	<u>Lesson 36 : Part 5 : TabIndex, TabStop, and New Forms (Default, Show, ShowDialog, Close)</u>
76	<u>Lesson 37 : Part 6 : MessageBox</u>
77	<u>Lesson 38 : Part 7: CheckBox, RadioButton, GroupBox, and Tag</u>

78	<u>Lesson 39 : Pizza Project (Solution)</u>
79	<u>Lesson 40 : More about TextBox</u>
80	<u>Lesson 41 : PictureBox</u>
81	<u>Lesson 42 & 43 : PictureBox Exercise & (Solution)</u>
81	<u>Lesson 44 : Draw Line , ellipse , and rectangle</u>
82	<u>Lesson 45 & 46 : Tic-Tac-Toe Game (Requirement / Solution)</u>
82	<u>Lesson 47 : Tic-Tac-Toe Game Solution One Event</u>
83	<u>Lesson 48 : MaskedTextBox</u>
84	<u>Lesson 49 : ComboBox</u>
85	<u>Lesson 50 : Combo Box Exercise / Solution</u>
86	<u>Lesson 51 : LinkLabel</u>
86	<u>Lesson 52 : CheckBoxList</u>
87	<u>Lesson 53 : DateTimePicker</u>
88	<u>Lesson 54 : MonthCalandar</u>
89	<u>Lesson 55 : Timer</u>
89	<u>Lesson 56 : NotifyIcon</u>
90	<u>Lesson 57 : TreeView and ImageList</u>
91	<u>Lesson 58 : ProgressBar</u>
92	<u>Lesson 59 : ListView</u>
93	<u>Lesson 60 : ErrorProvider</u>
93	<u>Lesson 61 : TrackBar</u>
94	<u>Lesson 62 : NumericUpDown</u>

94	<u>Lesson 63 : TabControl</u>
94	<u>Lesson 64 : GroupBox vs Panel</u>
95	<u>Lesson 65 : ColorDialog</u>
95	<u>Lesson 66 : FontDialog</u>
96	<u>Lesson 67 : Save File Dialog</u>
97	<u>Lesson 68 : Open File Dialog</u>
98	<u>Lesson 69 : Folder Browser Dialog</u>
98	<u>Lesson 70 : MDI Container</u>
99	<u>Lesson 71 : MenuStrip</u>
99	<u>Lesson 72 : Context Menu</u>