AMIT ACADEMY
for Computer Education

Subject Title: WEB DEVELOPMENTUSING ASP.NET
Subject Code: BCA 501

Unit 1: Introduction to Client Server Architecture

Prepared by Ankit Rami(AR) Any Query

Contact – +91 8460467193

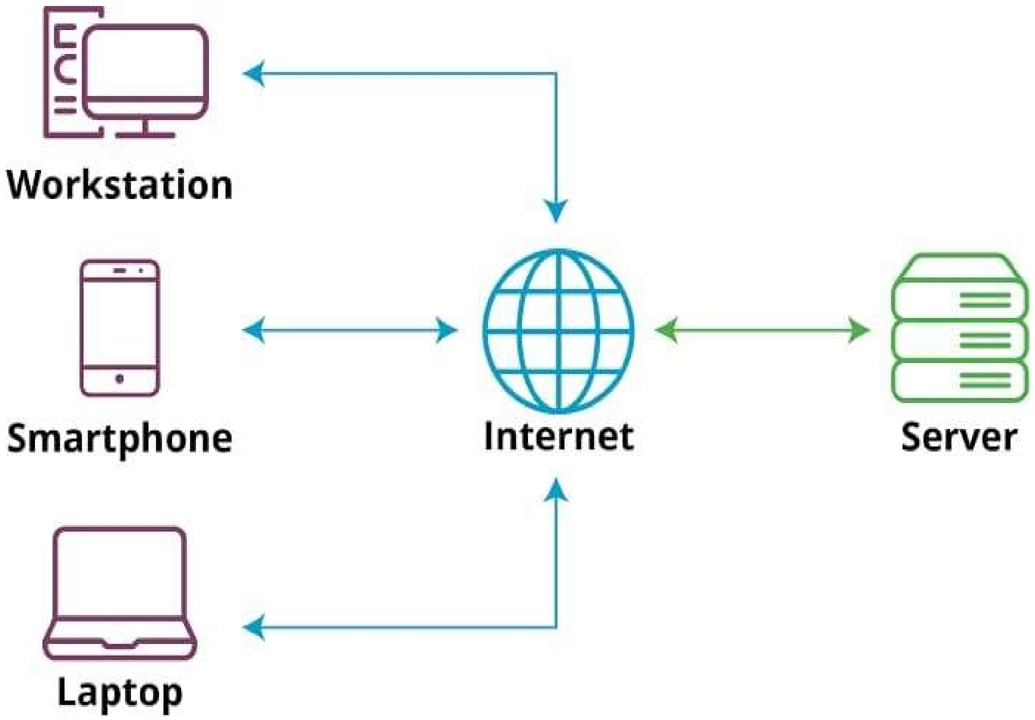
Email – ankitramiblog@gmail.com

What is Client-Server Architecture?

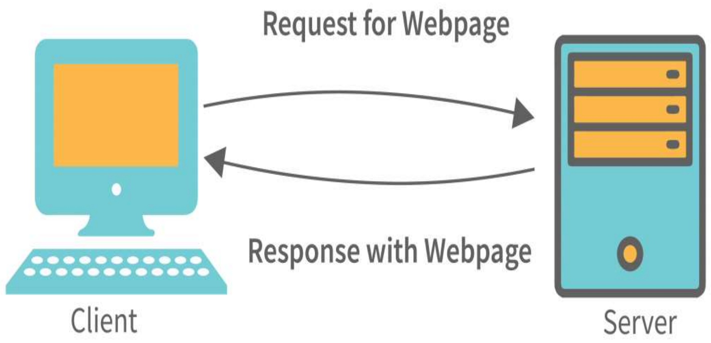
- The Client-server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters called clients.
- In the client-server architecture, when the client computer sends a request for data to the server through the internet, the server accepts the requested process and delivers the data packets requested back to the client.
- Clients do not share any of their resources.
- Examples of the Client-Server Model are Email, World Wide Web, etc.
- The client-server architecture refers to a system that hosts, delivers, and manages most of the resources and services that the client requests. In this model, all requests and services are delivered over a network, and it is also referred to as the networking computing model or client server network.
- Client-server architecture, alternatively called a client-server model, is a network application that breaks down tasks and workloads between clients and servers that reside on the same system or are linked by a computer network.

How Does the Client-Server Model Work?

- **Client:** When we say the word **Client**, it means to talk of a person or an organization using a particular service. Similarly in the digital world, a **Client** is a computer (**Host**) i.e. capable of receiving information or using a particular service from the service providers (**Servers**).
- **Servers:** Similarly, when we talk about the word **Servers**, It means a person or medium that serves something. Similarly in this digital world, a **Server** is a remote computer that provides information (data) or access to particular services.
- **First, the client sends their request via a network-enabled device**
- **Then, the network server accepts and processes the user request**
- **Finally, the server delivers the reply to the client**

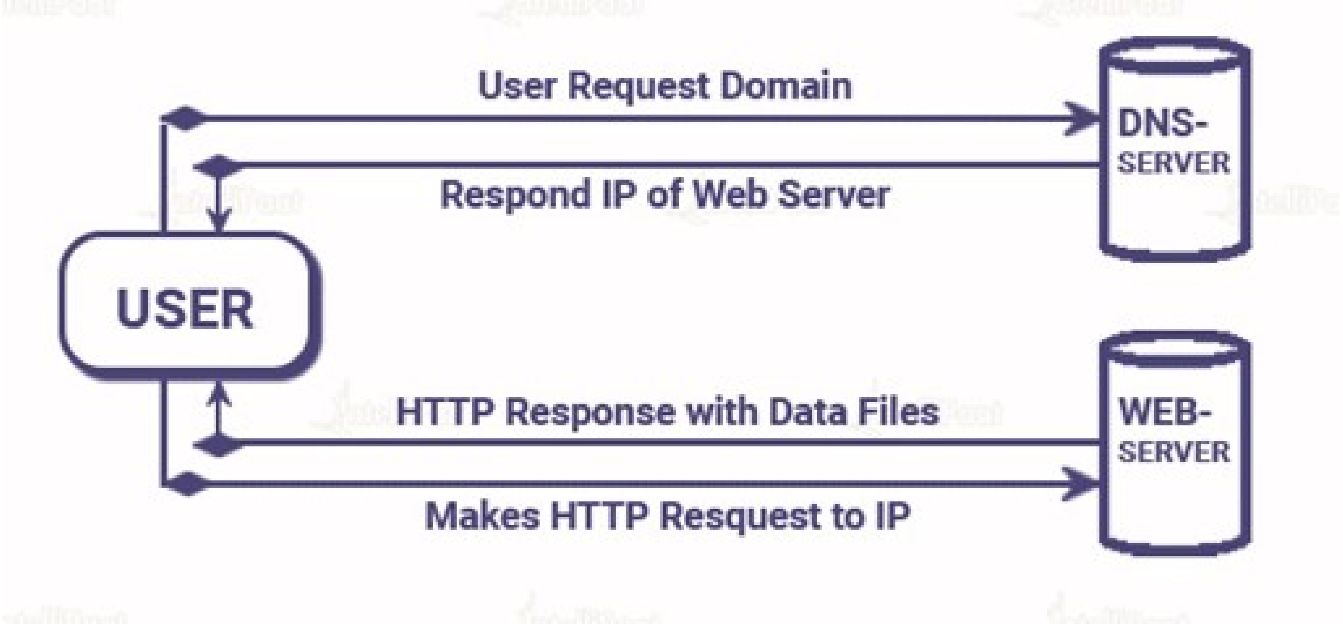


Client Send Request
Server Send Response

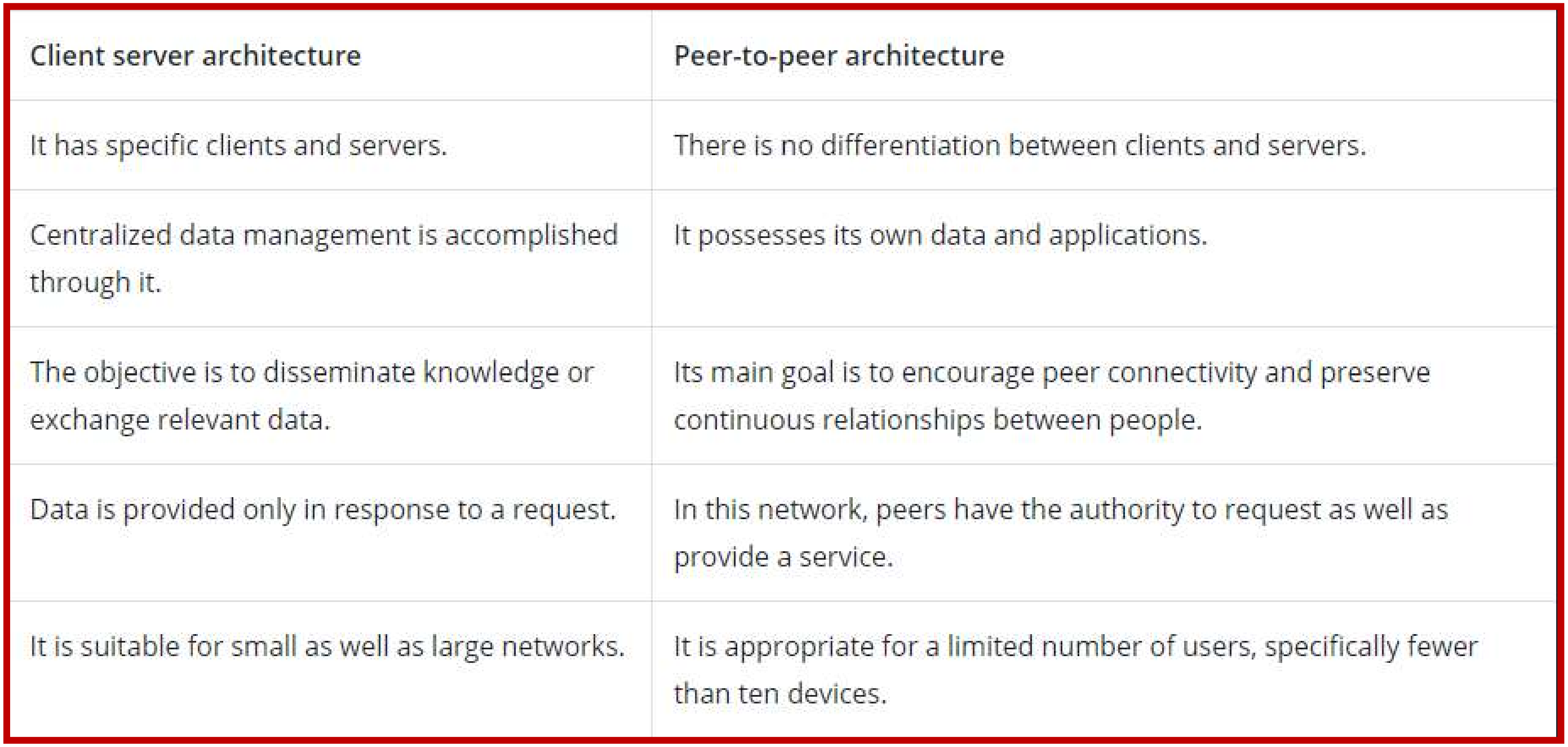


How Does the Client-Server Model Work?

- The user enters the uniform resource locator (URL) of the website or file and the browser sends a request to the domain name system (DNS) server.
- DNS server is responsible for searching and retrieving the IP address associated with a web server and then initiating actions using that IP address.
- After the DNS server responds, the browser sends over an HTTP or HTTPS request to the web server's IP, which was provided by the DNS server.
- Following the request, the server proceeds to transmit the essential website files required.
- Ultimately, the files are processed by the browser and the website is subsequently presented for viewing.

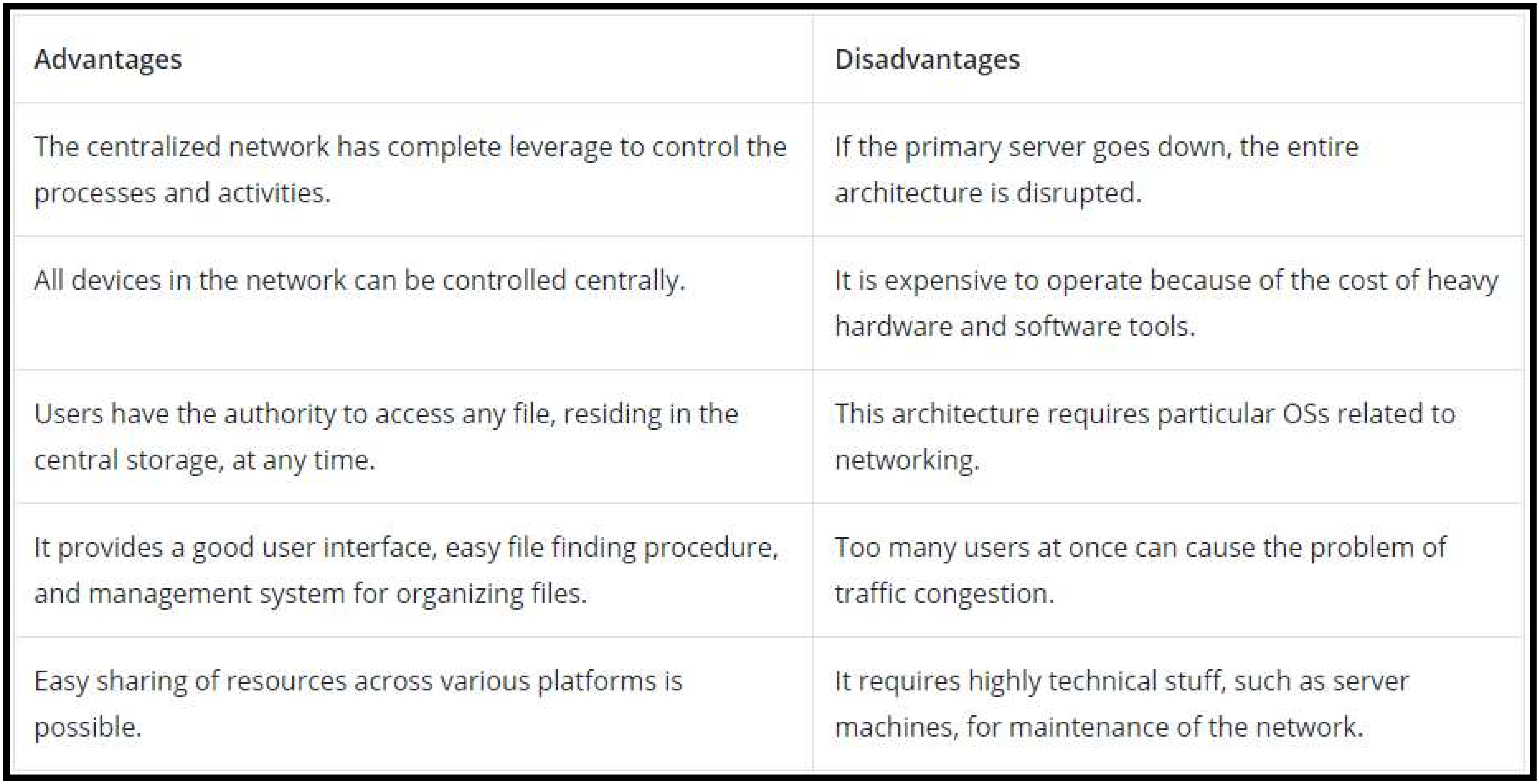


Difference between peer-to-peer network and Client-Server architecture?



Client server architecture	Peer-to-peer architecture
It has specific clients and servers.	There is no differentiation between clients and servers.
Centralized data management is accomplished through it.	It possesses its own data and applications.
The objective is to disseminate knowledge or exchange relevant data.	Its main goal is to encourage peer connectivity and preserve continuous relationships between people.
Data is provided only in response to a request.	In this network, peers have the authority to request as well as provide a service.
It is suitable for small as well as large networks.	It is appropriate for a limited number of users, specifically fewer than ten devices.

Advantages and disadvantages of Client-Server Architecture

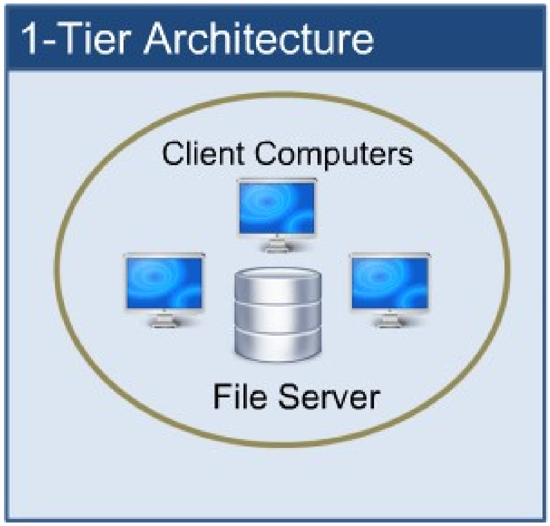


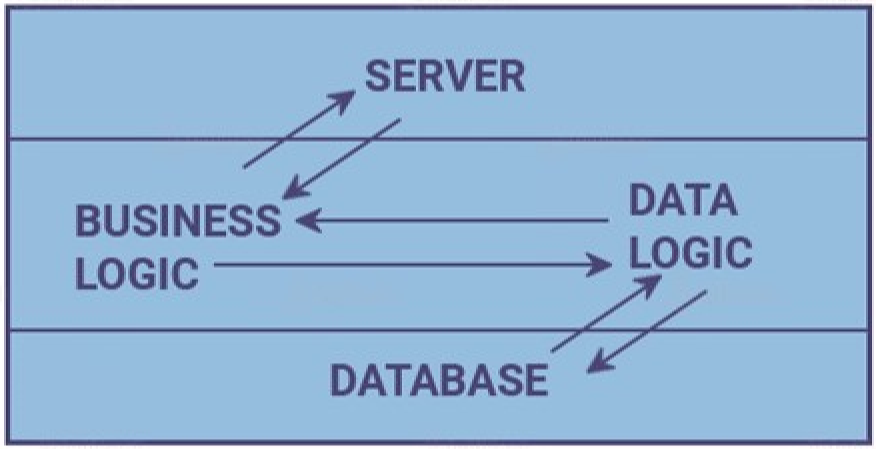
Advantages	Disadvantages
The centralized network has complete leverage to control the processes and activities.	If the primary server goes down, the entire architecture is disrupted.
All devices in the network can be controlled centrally.	It is expensive to operate because of the cost of heavy hardware and software tools.
Users have the authority to access any file, residing in the central storage, at any time.	This architecture requires particular OSs related to networking.
It provides a good user interface, easy file finding procedure, and management system for organizing files.	Too many users at once can cause the problem of traffic congestion.
Easy sharing of resources across various platforms is possible.	It requires highly technical stuff, such as server machines, for maintenance of the network.

Types of Client-Server Architecture

1-tier architecture

- In 1-tier architecture, the presentation, business logic, and data layers are all stored on a single device or a shared storage device. A good example is a desktop application that works offline and stores all its data on the same device it's running from.
- One-tier architecture has a Presentation layer, Business layer and Data layers at the same tier i.e. at the Client Tier. As the name suggested, all the layers and components are available on the same machine. MP3 players, MS Office etc. are some examples of one-tier architecture. To store the data (as a function of the Data Layer) local system or a shared drive is used.

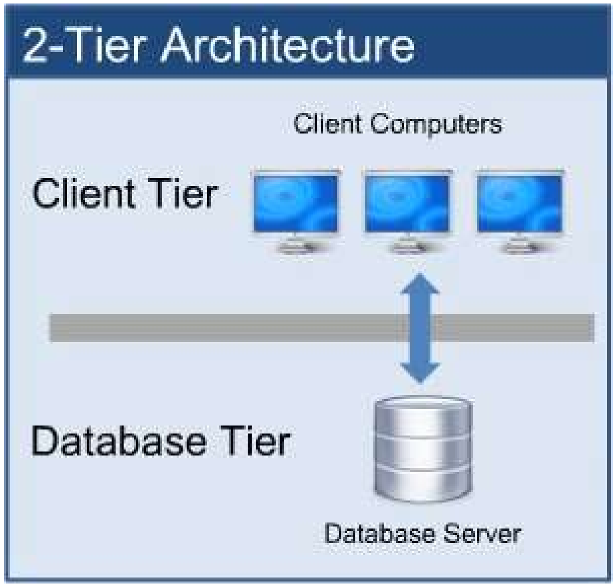


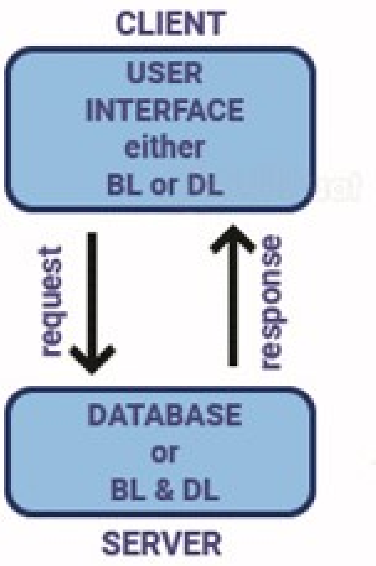


Types of Client-Server Architecture

2-tier architecture

- In 2-tier architecture, the presentation and business logic layers are stored on the client while the data layer is stored on a server. So as long as the code of an application is fully executed on the client and some of the data is being stored in a remote database, that application fits the 2-tier architecture criteria. A desktop application that requires you to log into an online account is a good example.
- In such type of architecture, the client tier handles both Presentation and Application layers and the server handles the Database layer. The two-tier architecture is also known as a ‘Client-Server Application’. In two-tier architecture, communication takes place between the Client and the Server. The client system sends the request to the server system and the server system processes the request and sends the response back to the client system.

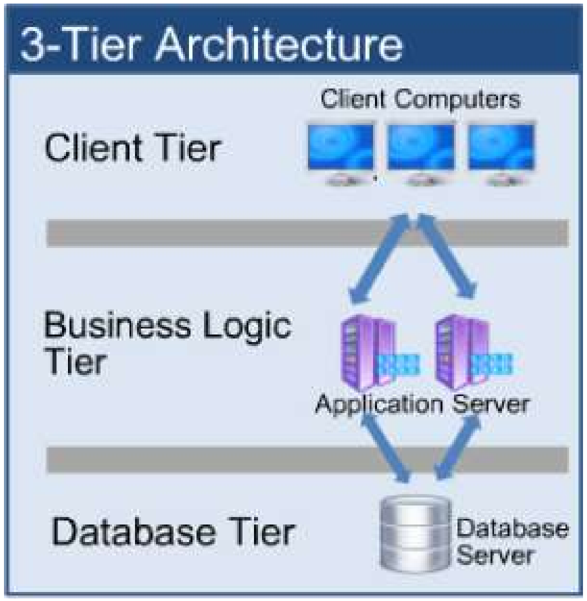


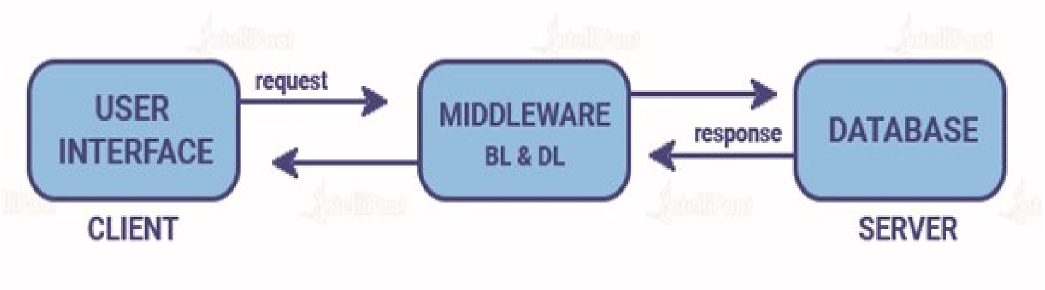


Types of Client-Server Architecture

3-tier architecture

- The presentation layer is stored on the client, the business logic layer is stored on one server, and the data layer is stored on another server. For example, you use a Smartphone app’s user interface to interact with an app while an application server executes most of the app’s code and a database server stores the data
- All three major layers are separated from each other. The presentation layer resides at Client Tier, the Application layer acts as middleware and lies at Business Tier and the Data layer is available at Data Tier. This is a very common architecture.

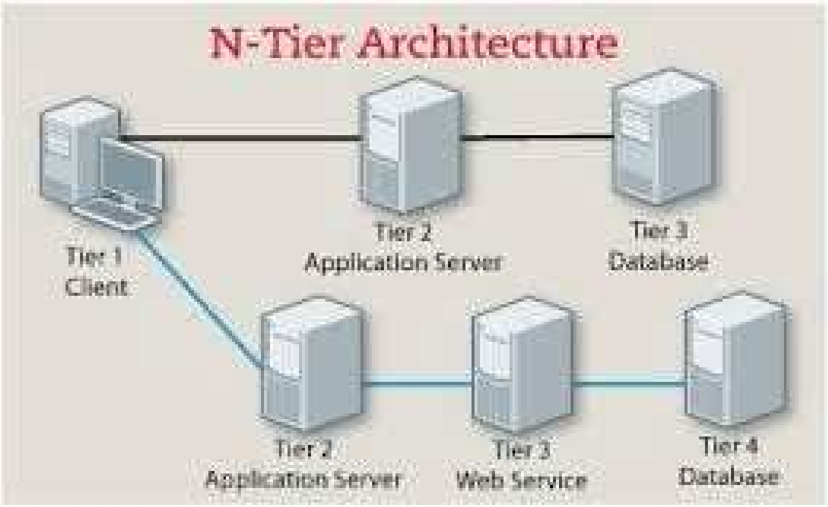


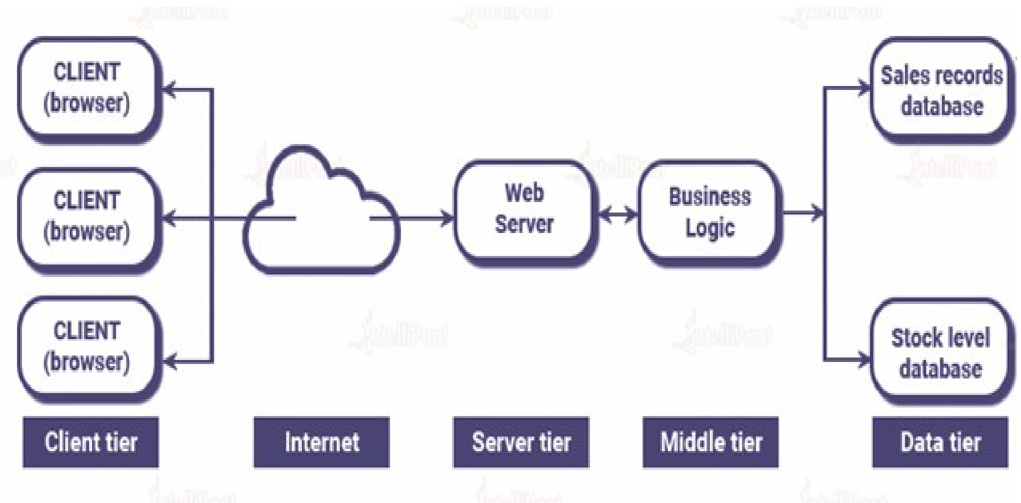


Types of Client-Server Architecture

N-tier architecture

- In N-tier architecture, the presentation and data layers are left untouched, compared to the 3-tier architecture. The difference is that N-tier architecture splits the business logic layer into multiple layers to improve performance, management, and stability.
- N-tier architecture is also called Distributed Architecture or Multi-tier Architecture. It is similar to three-tier architecture but the number of the application server is increased and represented in individual tiers in order to distribute the business logic so that the logic can be distributed.



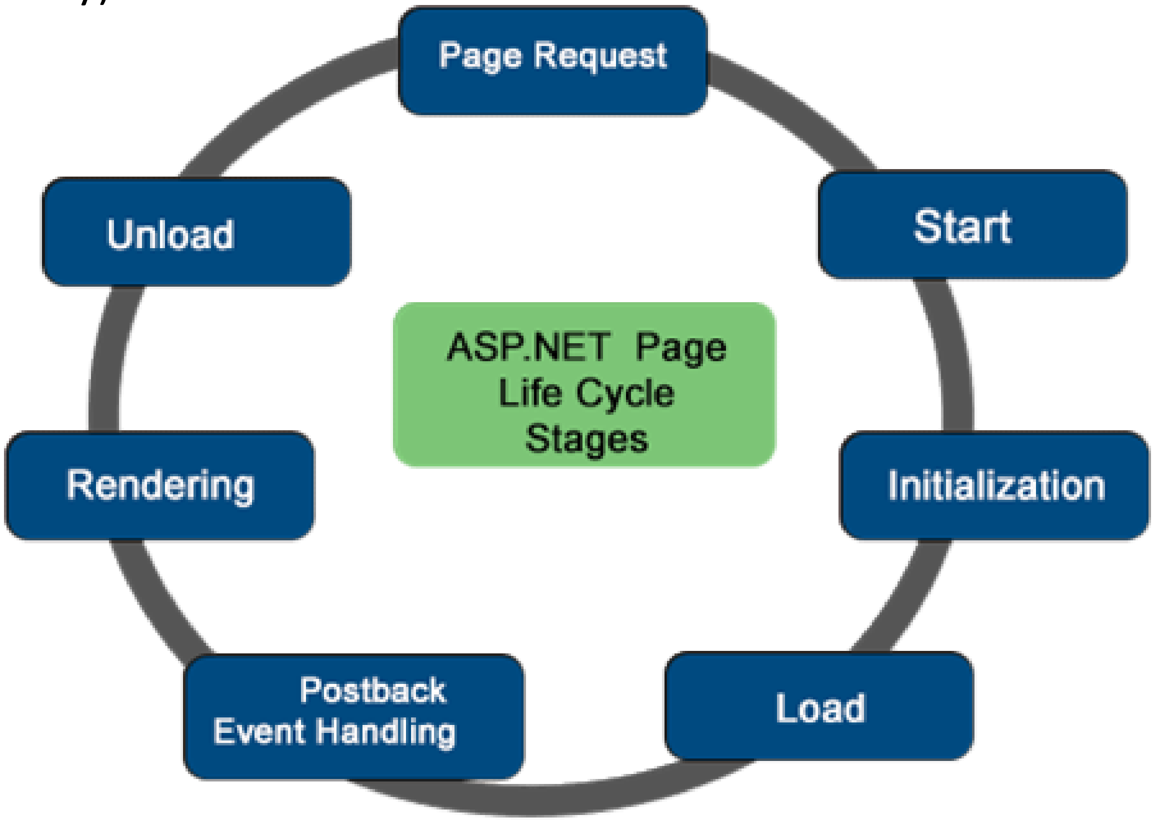


Introduction of ASP. Net

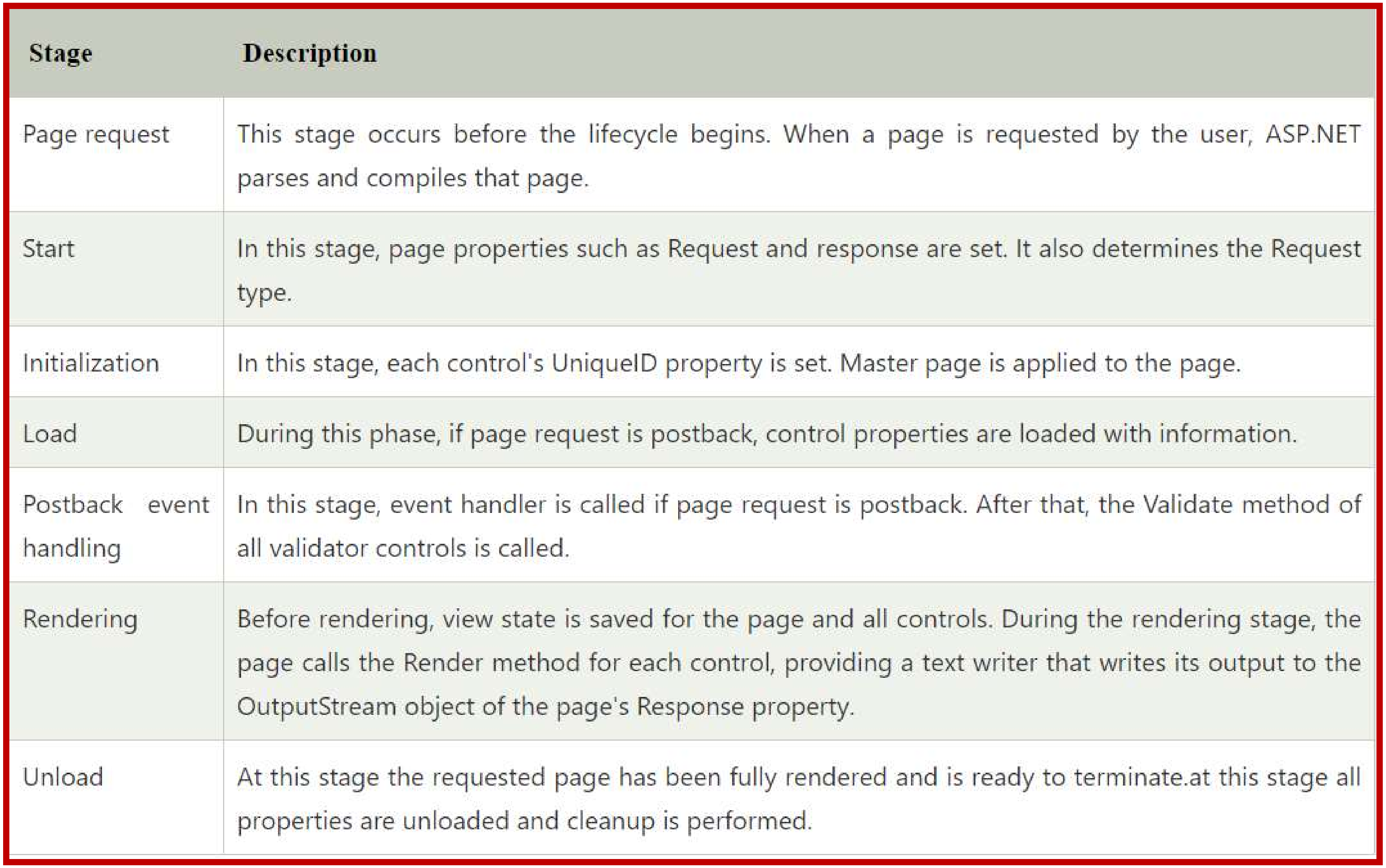
- **ASP stands for Active Server Pages**
- ASP and ASP.NET are server side technologies
- ASP pages have the file extension **.asp and .aspx** both are normally written in VBScript & C# Script.
- ASP.NET is a web framework designed and developed by Microsoft.
- It is used to develop websites, web applications and web services.
- It provides fantastic integration of HTML, CSS and JavaScript.
- It was first released in January 2002.
- It is built on the Common Language Runtime (CLR) and allows programmers to write code using any supported .NET language.
- .NET Framework is used to create a variety of applications and services like Console, Web, and Windows, etc. But ASP.NET is only used to create web applications and web services. That's why we termed ASP.NET as a subset of the .NET Framework.

What is Page Life Cycle in ASP.net?

- Every programmer should understand ASP.NET page life cycle because he should know that where we have to place particular methods and when we have to set page related properties. Additionally, if we are developing custom server controls then clear understanding of page life cycle is very useful to correct initialization of control, setting properties with view-state data, and control's code(Control events are subject to ASP.NET page only)

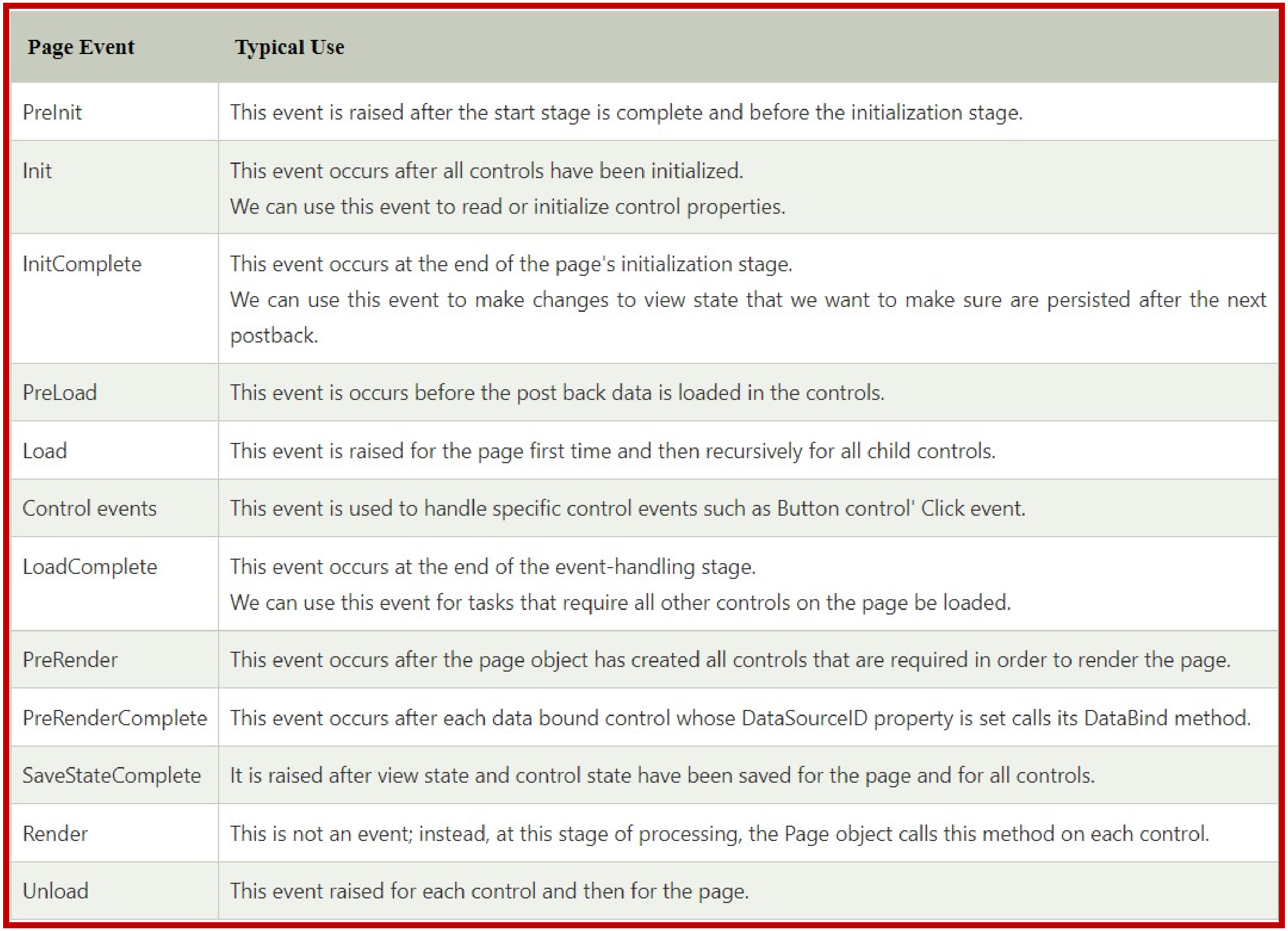


ASP.NET Page Life Cycle Stages



Stage	Description
Page request	This stage occurs before the lifecycle begins. When a page is requested by the user, ASP.NET parses and compiles that page.
Start	In this stage, page properties such as Request and response are set. It also determines the Request type.
Initialization	In this stage, each control's UniqueID property is set. Master page is applied to the page.
Load	During this phase, if page request is postback, control properties are loaded with information.
Postback event handling	In this stage, event handler is called if page request is postback. After that, the Validate method of all validator controls is called.
Rendering	Before rendering, view state is saved for the page and all controls. During the rendering stage, the page calls the Render method for each control, providing a text writer that writes its output to the OutputStream object of the page's Response property.
Unload	At this stage the requested page has been fully rendered and is ready to terminate.at this stage all properties are unloaded and cleanup is performed.

ASP.NET Page Life Cycle Events



Page Event	Typical Use
PreInit	This event is raised after the start stage is complete and before the initialization stage.
Init	This event occurs after all controls have been initialized. We can use this event to read or initialize control properties.
InitComplete	This event occurs at the end of the page's initialization stage. We can use this event to make changes to view state that we want to make sure are persisted after the next postback.
PreLoad	This event is occurs before the post back data is loaded in the controls.
Load	This event is raised for the page first time and then recursively for all child controls.
Control events	This event is used to handle specific control events such as Button control' Click event.
LoadComplete	This event occurs at the end of the event-handling stage. We can use this event for tasks that require all other controls on the page be loaded.
PreRender	This event occurs after the page object has created all controls that are required in order to render the page.
PreRenderComplete	This event occurs after each data bound control whose DataSourceID property is set calls its DataBind method.
SaveStateComplete	It is raised after view state and control state have been saved for the page and for all controls.
Render	This is not an event; instead, at this stage of processing, the Page object calls this method on each control.
Unload	This event raised for each control and then for the page.

ASP.NET – Directives

- ASP.NET directives are instructions to specify optional settings, such as registering a custom control and page language. These settings describe how the web forms (.aspx) or user controls (.ascx) pages are processed by the .Net framework.
- The syntax for declaring a directive is:

```
<%@ directive_name attribute=value [attribute=value] %>
```

Types of ASP.Net Directive

❑ Application Directive
❑ Assembly Directive
❑ Control Directive
❑ Implements Directive
❑ Import Directive
❑ Master Directive
❑ MasterType Directive
❑ OutputCache Directive
❑ Page Directive
❑ PreviousPageType Directive
❑ Reference Directive
❑ Register Directive

ASP.NET – Directives

(1) Application Directive

The Application directive defines application-specific attributes.

Ex - <%@ Application Language="C#" %>

(2) Assembly Directive

The Assembly directive links an assembly to the page or the application at parse time.

Ex - <%@ Assembly Name ="myassembly" %>

(3) The Control Directive

The control directive is used with the user controls and appears in the user control (.ascx) files.

Ex - <%@ Control Language="C#" EnableViewState="false" %>

(4) Implements Directive

The Implement directive indicates that the web page, master page or user control page must implement the specified .Net framework interface.

Ex - <%@ Implements Interface="interface_name" %>

(5) Import Directive

The Import directive imports a namespace into a web page, user control page of application. If the Import directive is specified in the global.asax file, then it is applied to the entire application.

Ex - <%@ namespace="System.Drawing" %>

ASP.NET – Directives

(6) Master Directive

The Master directive specifies a page file as being the mater page.

Ex - <%@ MasterPage Language="C#" AutoEventWireup="true" CodeFile="SiteMater.master.cs" Inherits="SiteMaster" %>

(7) MasterType Directive

The MasterType directive assigns a class name to the Master property of a page, to make it strongly typed.

Ex - <%@ MasterType attribute="value"[attribute="value" ...] %>

(8) OutputCache Directive

The OutputCache directive controls the output caching policies of a web page or a user control.

Ex - <%@ OutputCache Duration="15" VaryByParam="None" %>

(9) Page Directive

The Page directive defines the attributes specific to the page file for the page parser and the compiler.

Ex - <%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" Trace="true" %>

ASP.NET – Directives

(10) PreviousPageType Directive

The PreviousPageType directive assigns a class to a page, so that the page is strongly typed.

Ex - <%@ PreviousPageType attribute="value"[attribute="value" ...] %>

(11) Reference Directive

The Reference directive indicates that another page or user control should be compiled and linked to the current page.

Ex - <%@ Reference Page ="somepage.aspx" %>

(12) Register Directive

The Register derivative is used for registering the custom server controls and user controls.

Ex - <%@ Register Src="~/footer.ascx" TagName="footer" TagPrefix="Tfooter" %>

What is C# (C-Sharp) Programming Language?

- ✓ C# (C-Sharp) is a programming language developed by Microsoft that runs on the .NET Framework.
- ✓ C# is used to develop web apps, desktop apps, mobile apps, games and much more.
- ✓ C# is a general-purpose, modern and object-oriented programming language pronounced as **“C sharp”**.
- ✓ It was developed by Microsoft led by **Anders Hejlsberg** and his team within the .NET initiative and was approved by the European Computer Manufacturers Association (ECMA) and International Standards Organization (ISO).

- ✓ C# is among the languages for Common Language Infrastructure and the current version of C# is version 7.2.

Why Use C# (C-Sharp) Programming Language?

- ✓ It is one of the most popular programming languages in the world
- ✓ It is easy to learn and simple to use
- ✓ It has huge community support
- ✓ C# is an object-oriented language which gives a clear structure to programs and allows code to be reused, lowering development costs
- ✓ As C# is close to C, C++ and Java, it makes it easy for programmers to switch to C# or vice versa

C# (C-Sharp) Comments

- ✓ Comments can be used to explain C# code, and to make it more readable. It can also be used to prevent execution when testing alternative code.

Single-line Comments

- ✓ Single-line comments start with two forward slashes (//)
- ✓ Any text between // and the end of the line is ignored by C# **(will not be executed)**
- ✓ **EX –**

```
//Hello Ankit Rami
Response.Write(“Learn ASP.NET with Ankit Rami”);
```

Multi-line Comments

- ✓ Multi-line comments start with /* and ends with */
- ✓ Any text between /* and */ will be ignored by C# **(will not be executed)**
- ✓ **EX –**

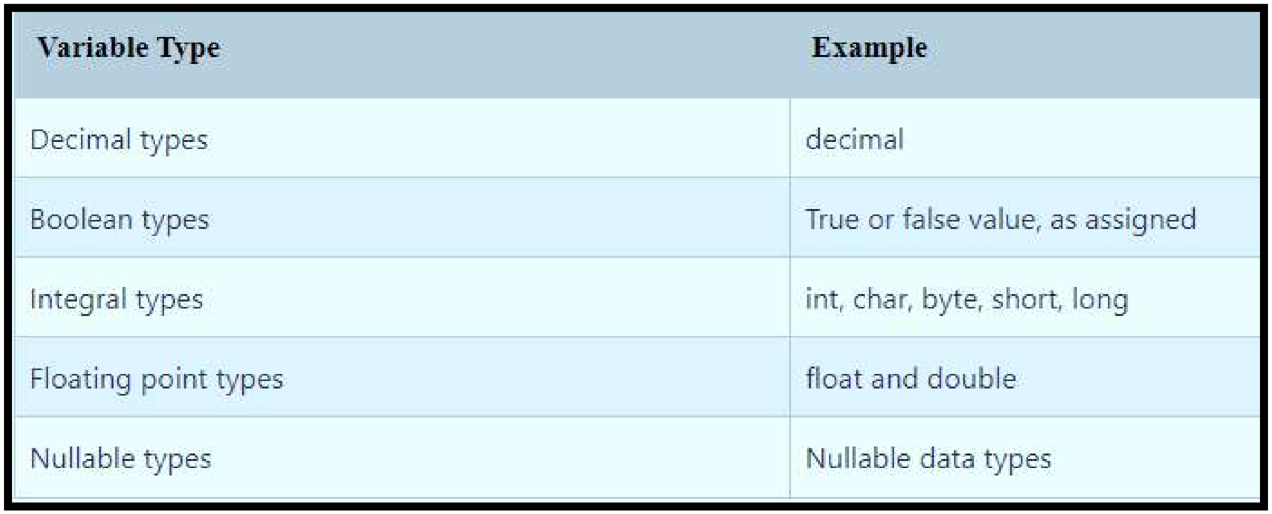
```
/* Hello Ankit Rami
   Welcome to ASP.NET Programming */
Response.Write(“Learn ASP.NET with Ankit Rami”);
```

C# (C-Sharp) Variables

- ✓ A variable is a name of memory location.
- ✓ Variables is used to store data.
- ✓ Variables value can be changed and it can be reused many times.
- ✓ It is a way to represent memory location through symbol so that it can be easily identified.
- ✓ Ex- a1, a_1, _a, b1 etc…

Rules for defining variables

- ✓ A variable can have alphabets, digits and underscore.
- ✓ A variable name can start with alphabet and underscore only. It can't start with digit.
- ✓ No white space is allowed within variable name.



Variable Type	Example
Decimal types	decimal
Boolean types	True or false value, as assigned
Integral types	int, char, byte, short, long
Floating point types	float and double
Nullable types	Nullable data types

C# (C-Sharp) Data Types

✓ A data type specifies the type of data that a variable can store such as integer, floating, character etc.

Different Data Types

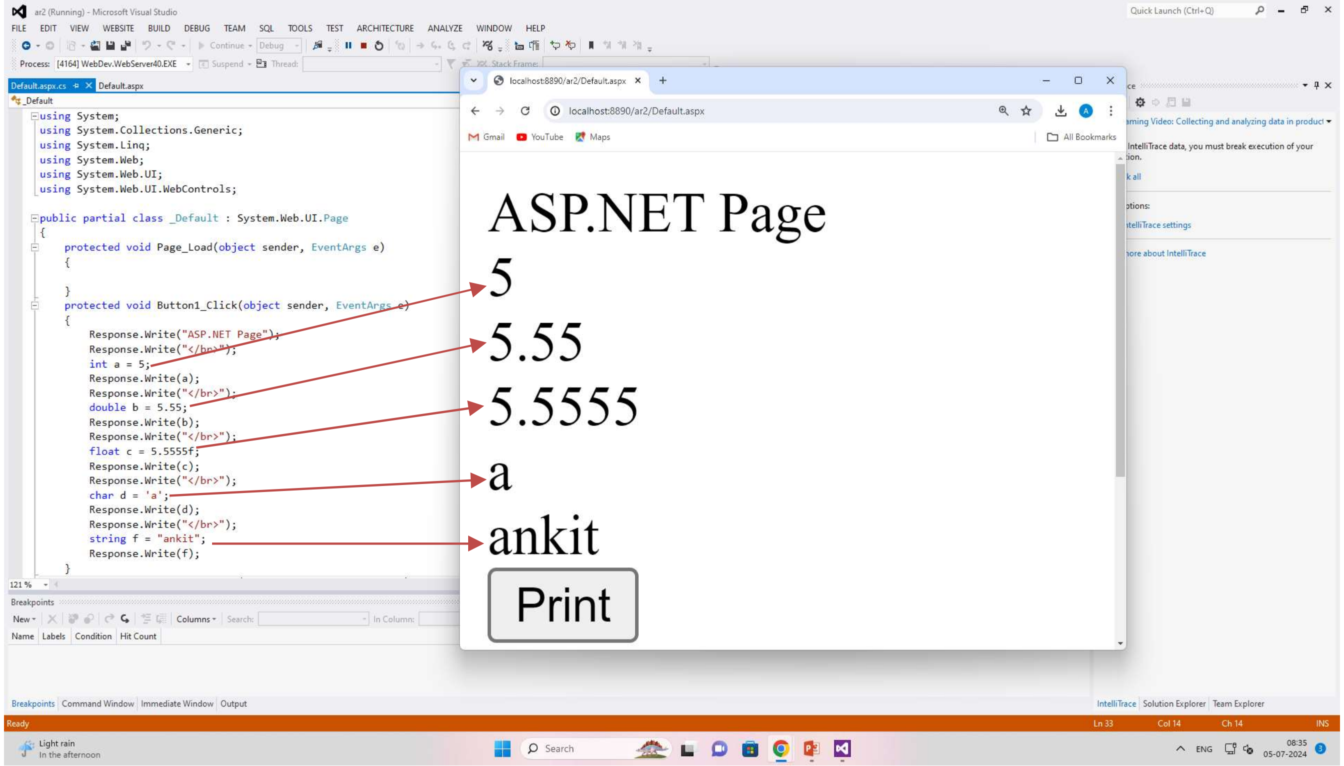
✓ **int** - stores integers (whole numbers), without decimals, such as 123 or -123

✓ **double** - stores floating point numbers, with decimals, such as 19.99 or -19.99

✓ **char** - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes

✓ **string** - stores text, such as "Hello World". String values are surrounded by double quotes

✓ **bool** - stores values with two states: true or false



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        Response.Write("ASP.NET Page");
        Response.Write("</br>");
        int a = 5;
        Response.Write(a);
        Response.Write("</br>");
        double b = 5.55;
        Response.Write(b);
        Response.Write("</br>");
        float c = 5.5555f;
        Response.Write(c);
        Response.Write("</br>");
        char d = 'a';
        Response.Write(d);
        Response.Write("</br>");
        string f = "ankit";
        Response.Write(f);
    }
```

```
ASP.NET Page
5
5.55
5.5555
a
ankit
```

ar2 (Running) - Microsoft Visual Studio
FILE EDIT VIEW WEBSITE BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Default.aspx.cs
Default.aspx
localhost:8890/ar2/Default.aspx
Print
Breakpoints
Command Window
Immediate Window
Output
IntelliTrace
Solution Explorer
Team Explorer
Ready

C# (C-Sharp) Control Statements

✓ In C#, a control statement is a programming construct that allows you to control the flow of execution in your code based on certain conditions. They enable decision-making and branching, ensuring that different parts of your code are executed based on specific criteria.

C# offers three types of control statements:

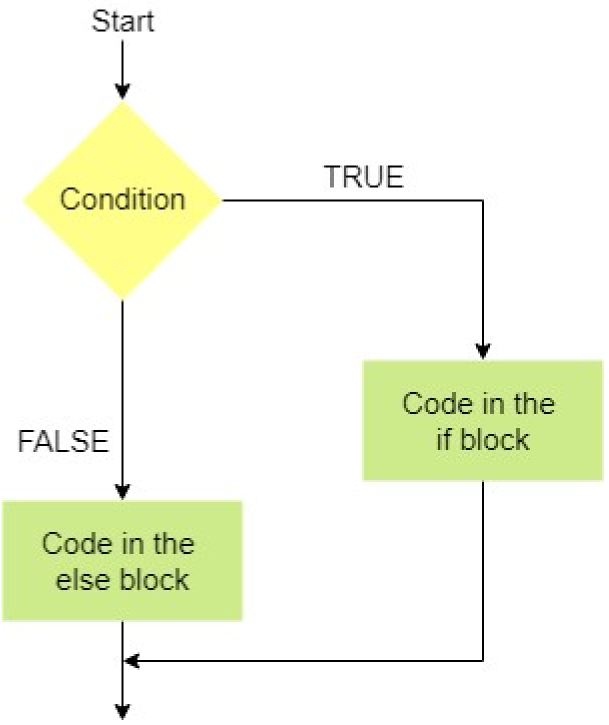
✓ **Selection Statements – This consists of if, else, else if, nested if, and switch**

✓ **Iteration Statements – This consists of for, foreach, while, and do while looping.**

✓ **Jump Statements – This consists of break, continue, return, and goto statements.**

C# (C-Sharp) Selection Control Statements (if, else, else if, and nested if)

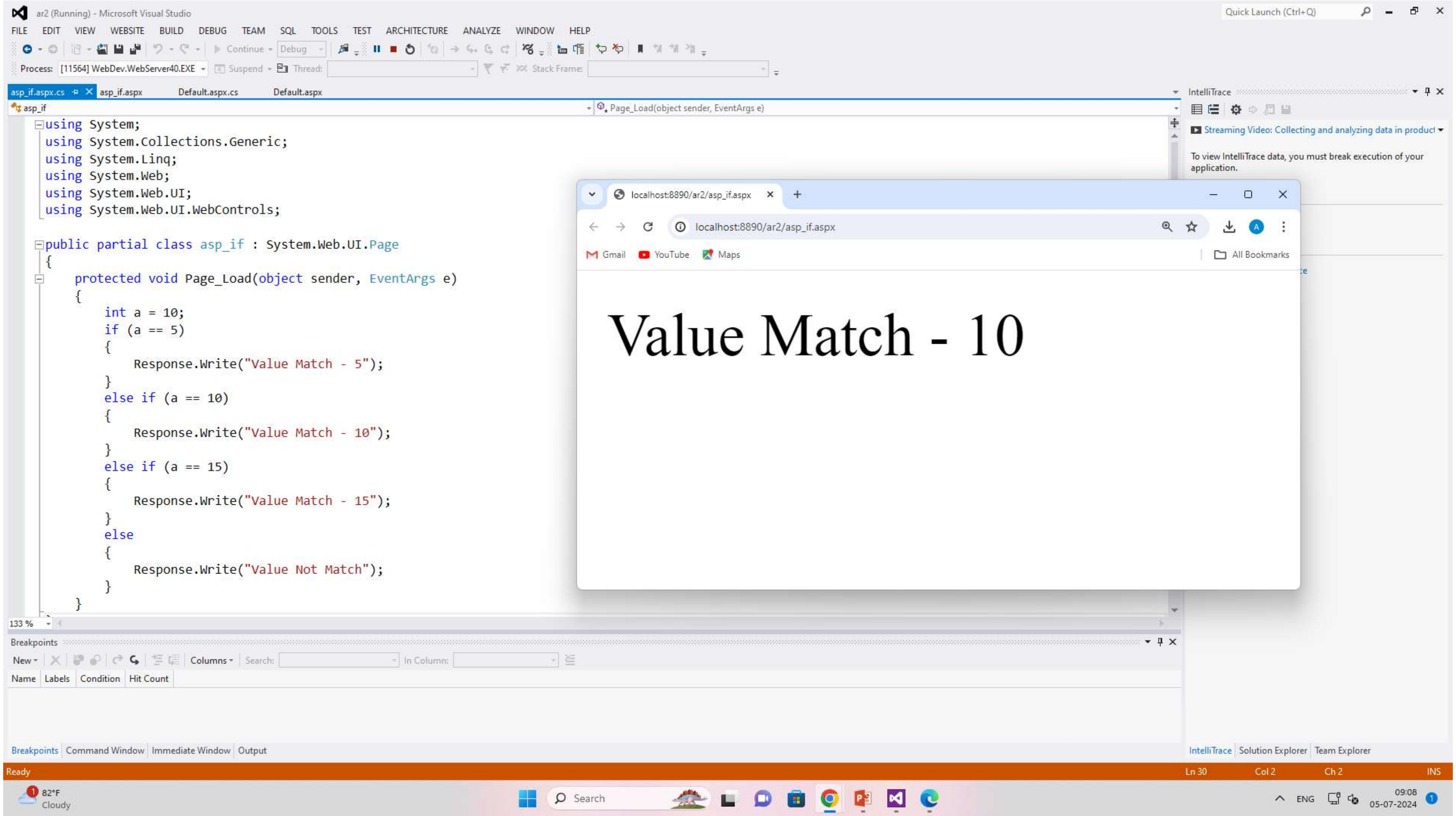
- ✓ The if statement is a fundamental control statement in C#. It evaluates a condition and executes a block of code if the condition is true. It can also be extended with an optional else clause to handle the case when the condition is false.



Syntax :

```
if (statement)
{
                statement;
}
else
{
                statement;
}
```

C# (C-Sharp) Selection Control Statements (if, else, else if, and nested if)


ar2 (Running) - Microsoft Visual Studio
FILE EDIT VIEW WEBSITE BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Process: [11564] WebDev.WebServer40.EXE
asp_if.aspx.cs
asp_if.aspx
Default.aspx.cs
Default.aspx
Page_Load(object sender, EventArgs e)
IntelliTrace
localhost:8890/ar2/asp_if.aspx
Breakpoints
Ready
Ln 30 Col 2 Ch 2 INS

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class asp_if : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        int a = 10;
        if (a == 5)
        {
            Response.Write("Value Match - 5");
        }
        else if (a == 10)
        {
            Response.Write("Value Match - 10");
        }
        else if (a == 15)
        {
            Response.Write("Value Match - 15");
        }
        else
        {
            Response.Write("Value Not Match");
        }
    }
```

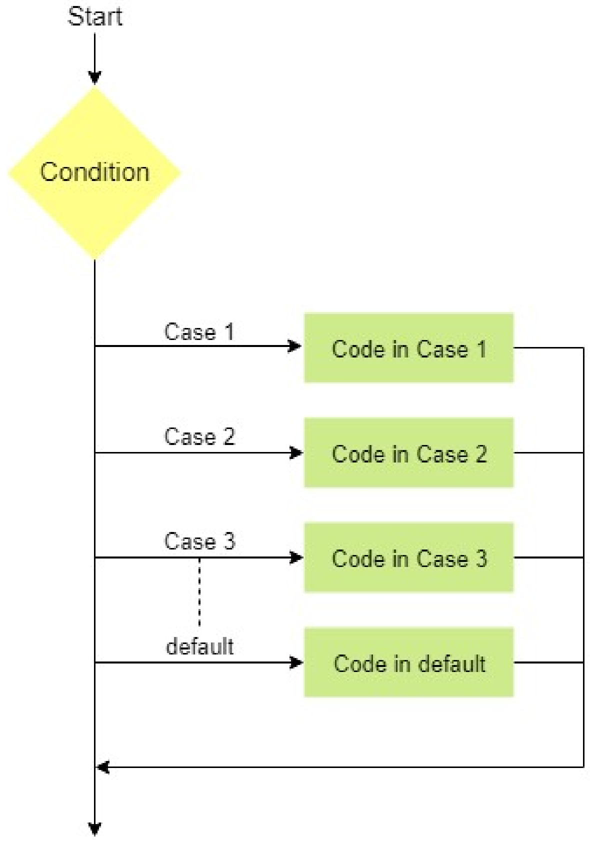
```
Value Match - 10
```

Prepared by Ankit Rami(AR) Any Query Contact – 8460467193 Email – ankitramiblog@gmail.com

C# (C-Sharp) Selection Control Statements

Switch Case

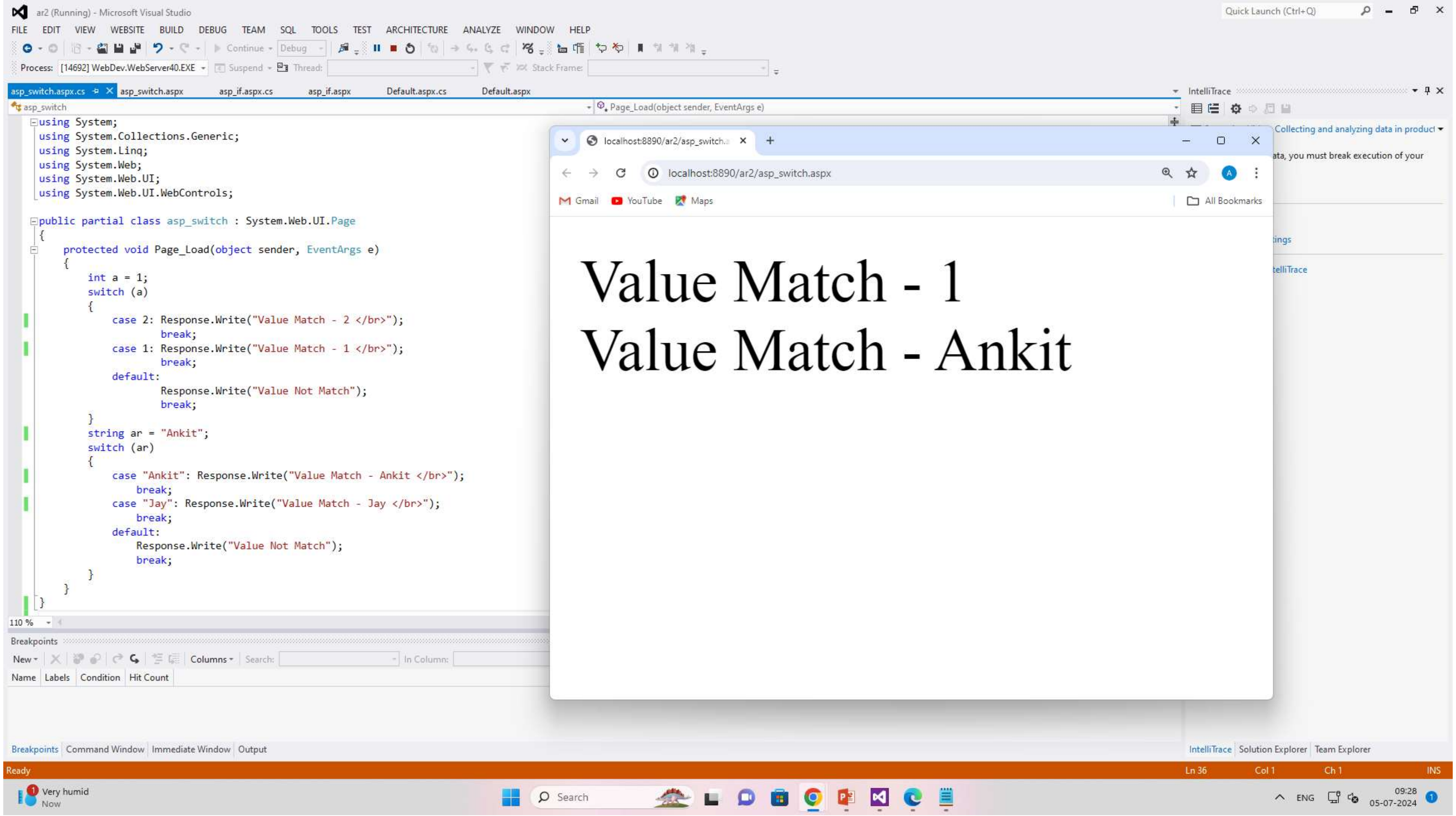
✓ The switch statement executes only one statement from multiple given statements associated with conditions. If any condition returns true, then the code statements below it get executed.



Syntax :

```
datatype Variable;
switch (Variable)
{
            case a:
                        statement;
                        break;
            case b:
                        statement;
                        break;
            default:
                        statement;
                        break;
}
```

C# (C-Sharp) Selection Control Statements Switch Case


ar2 (Running) - Microsoft Visual Studio
FILE EDIT VIEW WEBSITE BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Process: [14692] WebDev.WebServer40.EXE
asp_switch.aspx.cs
asp_switch.aspx
asp_if.aspx.cs
asp_if.aspx
Default.aspx.cs
Default.aspx
Page_Load(object sender, EventArgs e)
localhost:8890/ar2/asp_switch.aspx
Breakpoints
Ready
Ln 36
Col 1
Ch 1

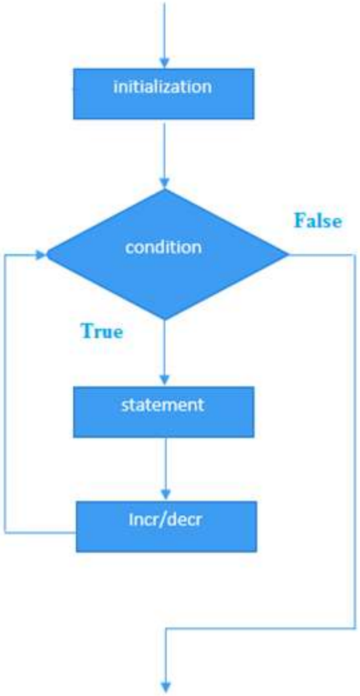
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class asp_switch : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        int a = 1;
        switch (a)
        {
            case 2: Response.Write("Value Match - 2 </br>");
                    break;
            case 1: Response.Write("Value Match - 1 </br>");
                    break;
            default:
                    Response.Write("Value Not Match");
                    break;
        }
        string ar = "Ankit";
        switch (ar)
        {
            case "Ankit": Response.Write("Value Match - Ankit </br>");
                break;
            case "Jay": Response.Write("Value Match - Jay </br>");
                break;
            default:
                Response.Write("Value Not Match");
                break;
        }
    }
}
```

```
Value Match - 1
Value Match - Ankit
```

C# (C-Sharp) Iteration Control Statements

For Loop

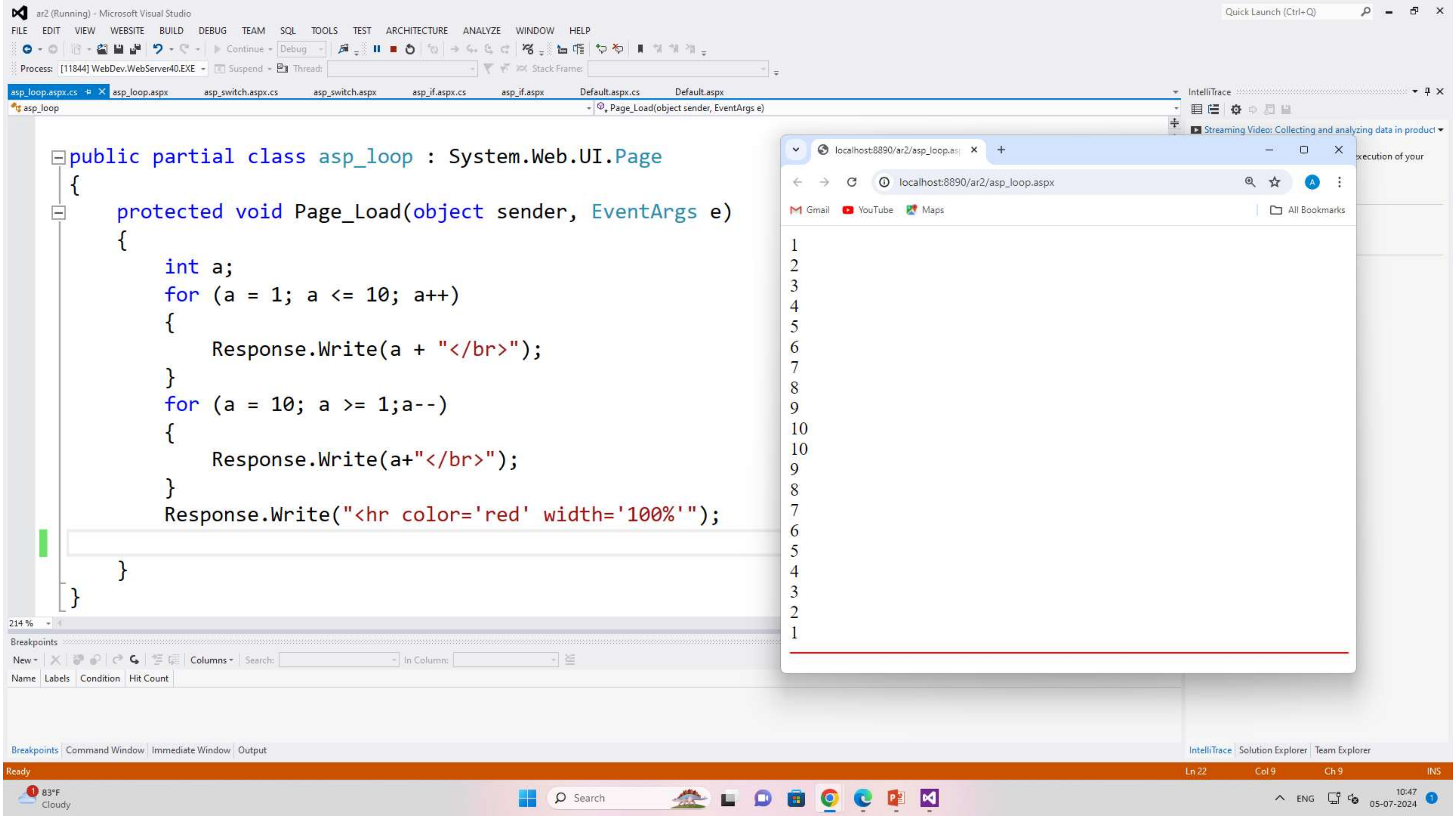
- ✓ The C# *for loop* is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop than while or do-while loops.
- ✓ The C# for loop is same as C/C++. We can initialize variable, check condition and increment/decrement value..



Syntax :

```
for(initialization; condition; incr/decr)
{
            //code to be executed
}
```

C# (C-Sharp) Iteration Control Statements For Loop



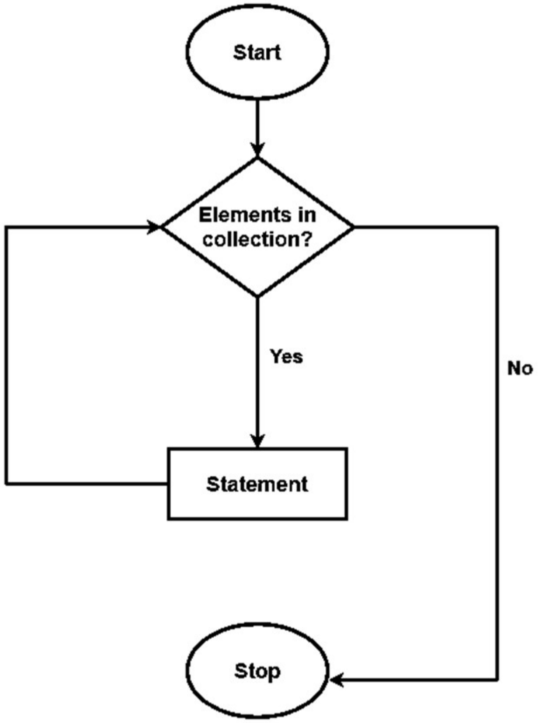
```
public partial class asp_loop : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        int a;
        for (a = 1; a <= 10; a++)
        {
            Response.Write(a + "</br>");
        }
        for (a = 10; a >= 1;a--)
        {
            Response.Write(a+"</br>");
        }
        Response.Write("<hr color='red' width='100%'");

    }
}
```

```
1
2
3
4
5
6
7
8
9
10
10
9
8
7
6
5
4
3
2
1
```

C# (C-Sharp) Iteration Control Statements

For Each Loop

✓ The foreach loop in C# iterates items in a collection, like an array or a list. It proves useful for traversing through each element in the collection and displaying them. The foreach loop is an easier and more readable alternative to for loop.

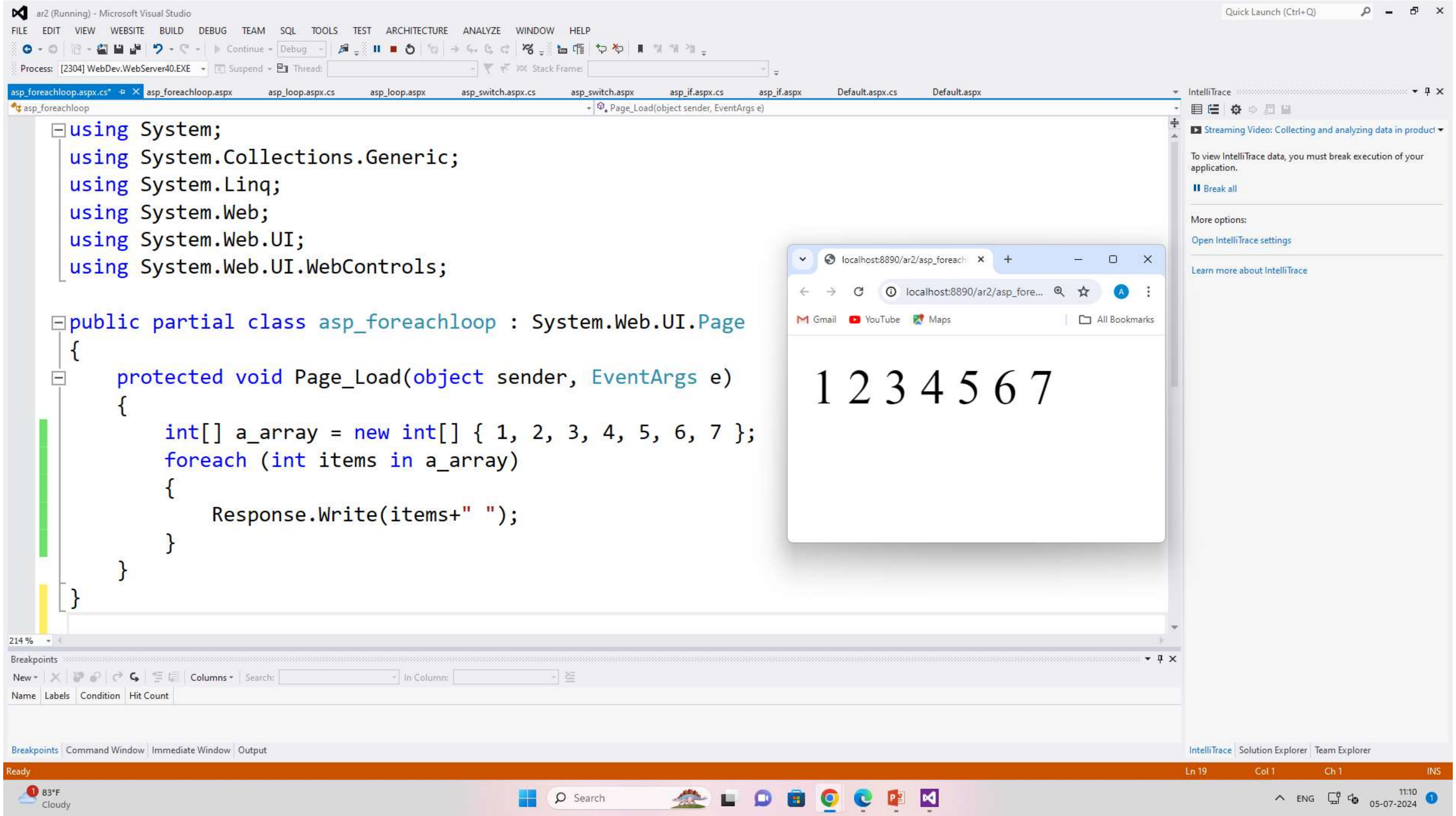


Syntax :

```
foreach(data_type var_name in collection_variable)
{
    // statements to be executed
}
```

C# (C-Sharp) Iteration Control Statements

For Each Loop



ar2 (Running) - Microsoft Visual Studio
FILE EDIT VIEW WEBSITE BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Process: [2304] WebDev.WebServer40.EXE
asp_foreachloop.aspx.cs*
asp_foreachloop.aspx
asp_loop.aspx.cs
asp_loop.aspx
asp_switch.aspx.cs
asp_switch.aspx
asp_if.aspx.cs
asp_if.aspx
Default.aspx.cs
Default.aspx
asp_foreachloop
Page_Load(object sender, EventArgs e)
IntelliTrace
Streaming Video: Collecting and analyzing data in product
To view IntelliTrace data, you must break execution of your application.
Break all
More options:
Open IntelliTrace settings
Learn more about IntelliTrace
localhost:8890/ar2/asp_foreach
localhost:8890/ar2/asp_fore...
Gmail
YouTube
Maps
All Bookmarks
Breakpoints
Name Labels Condition Hit Count
Breakpoints Command Window Immediate Window Output
IntelliTrace Solution Explorer Team Explorer
Ready
Ln 19 Col 1 Ch 1 INS
83°F Cloudy
Search
ENG
11:10
05-07-2024

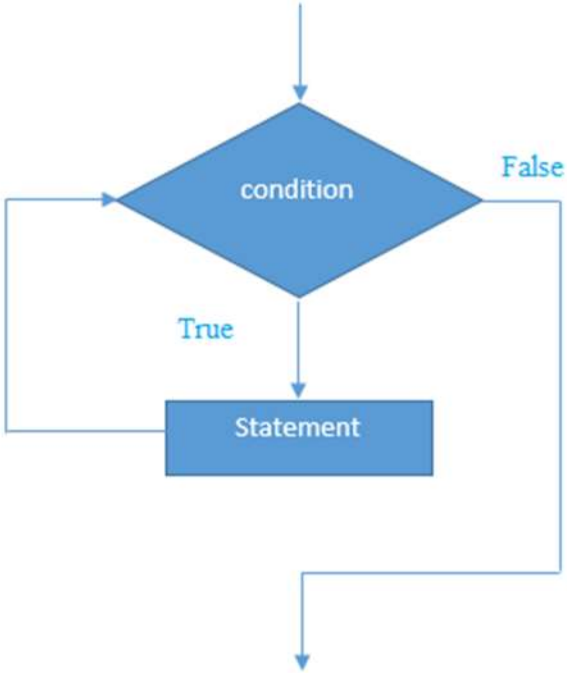
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class asp_foreachloop : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        int[] a_array = new int[] { 1, 2, 3, 4, 5, 6, 7 };
        foreach (int items in a_array)
        {
            Response.Write(items+" ");
        }
    }
}
```

```
1 2 3 4 5 6 7
```

C# (C-Sharp) Iteration Control Statements

While Loop

✓ In C#, *while loop* is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop than for loop.

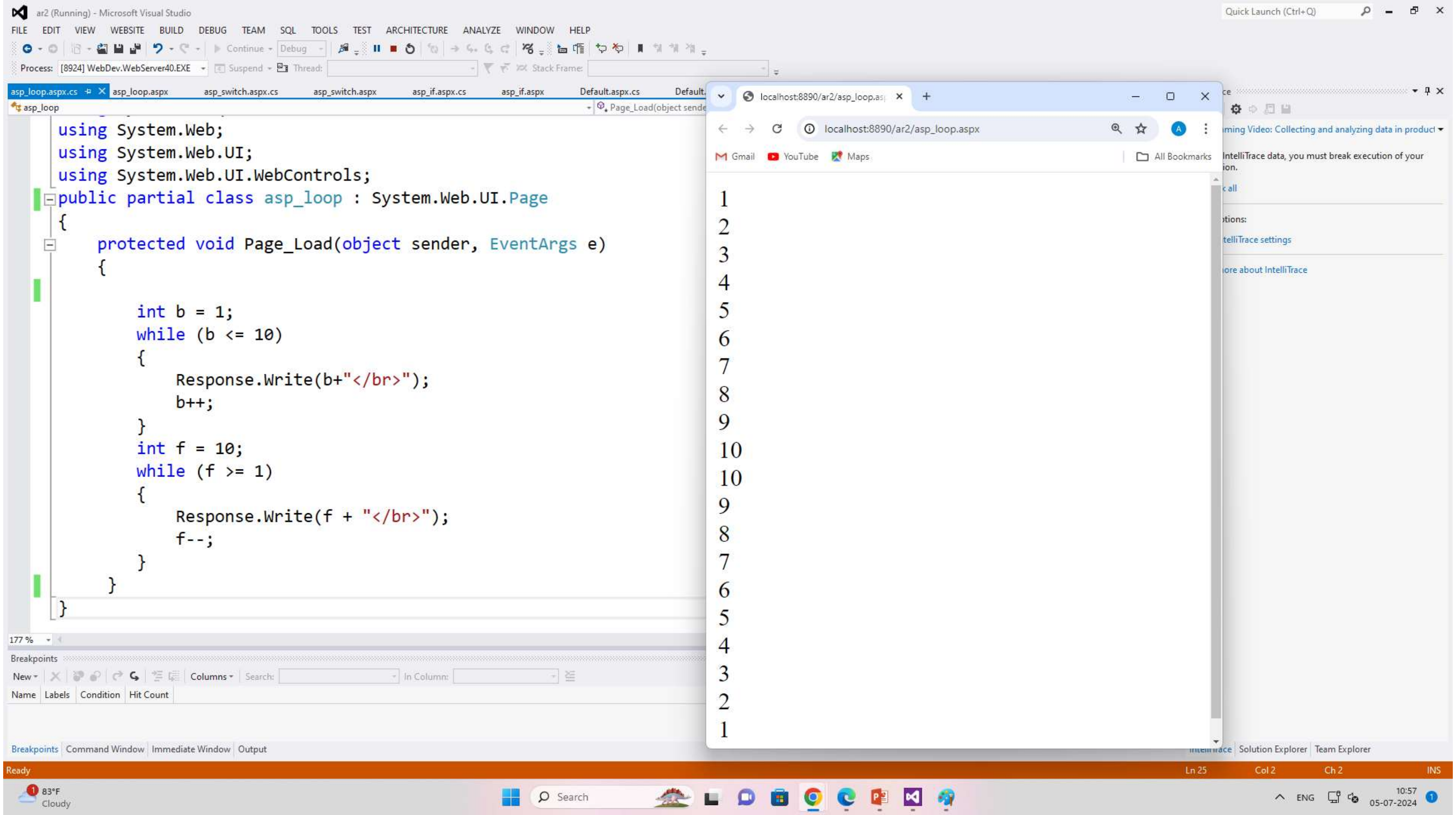


Syntax :

```
while(condition)
{
            //code to be executed
}
```

C# (C-Sharp) Iteration Control Statements

While Loop



```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
public partial class asp_loop : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

        int b = 1;
        while (b <= 10)
        {
            Response.Write(b+"</br>");
            b++;
        }
        int f = 10;
        while (f >= 1)
        {
            Response.Write(f + "</br>");
            f--;
        }
    }
}
```

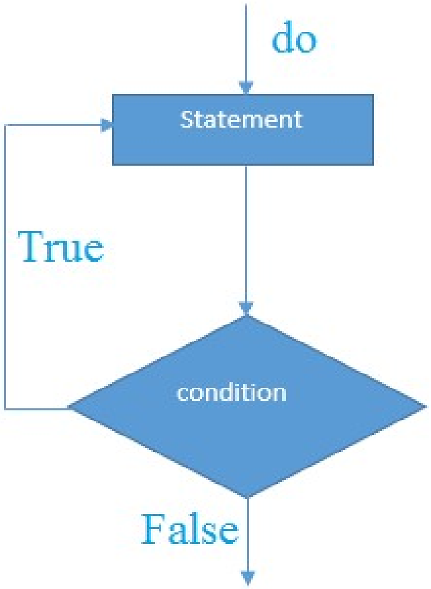
localhost:8890/ar2/asp_loop.aspx

```
1
2
3
4
5
6
7
8
9
10
10
9
8
7
6
5
4
3
2
1
```

C# (C-Sharp) Iteration Control Statements

Do While Loop

- ✓ The C# *do-while loop* is used to iterate a part of the program several times. If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use do-while loop.
- ✓ The C# *do-while loop* is executed at least once because condition is checked after loop body.

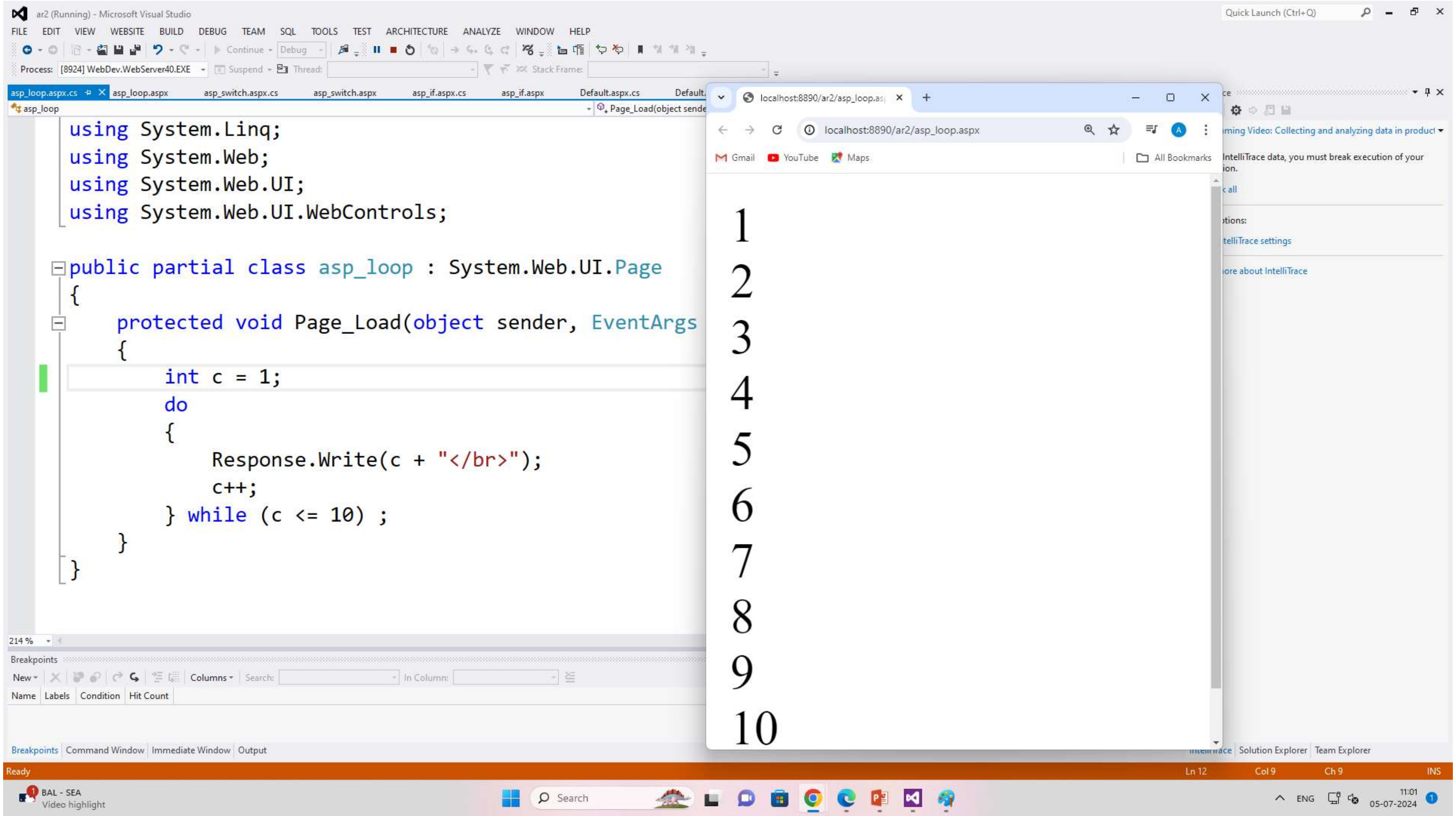


Syntax :

```
do
{
        //code to be executed
}while(condition);
```

C# (C-Sharp) Iteration Control Statements

Do While Loop



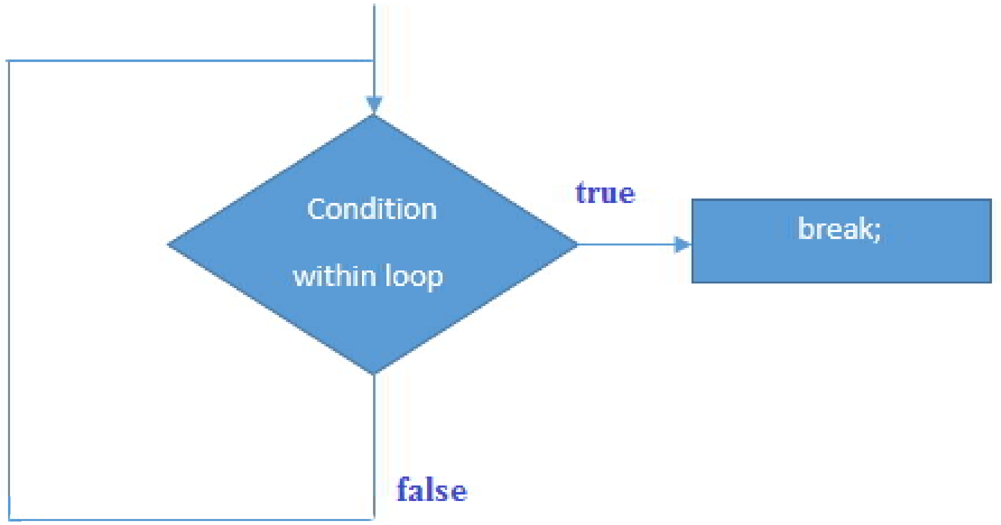
ar2 (Running) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW WEBSITE BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Continue Debug
Process: [8924] WebDev.WebServer40.EXE Suspend Thread: Stack Frame:
asp_loop.aspx.cs asp_loop.aspx asp_switch.aspx.cs asp_switch.aspx asp_if.aspx.cs asp_if.aspx Default.aspx.cs Default.
asp_loop
Page_Load(object sende
localhost:8890/ar2/asp_loop.as
localhost:8890/ar2/asp_loop.aspx
Gmail YouTube Maps All Bookmarks
ming Video: Collecting and analyzing data in product
IntelliTrace data, you must break execution of your ion.
all
tions:
telliTrace settings
ore about IntelliTrace
214 %
Breakpoints
New Columns Search: In Column:
Name Labels Condition Hit Count
Breakpoints Command Window Immediate Window Output
IntelliTrace Solution Explorer Team Explorer
Ready Ln 12 Col 9 Ch 9 INS
BAL - SEA
Video highlight
Search
ENG 11:01 05-07-2024

```
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class asp_loop : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs
    {
        int c = 1;
        do
        {
            Response.Write(c + "</br>");
            c++;
        } while (c <= 10) ;
    }
}
```

```
1
2
3
4
5
6
7
8
9
10
```

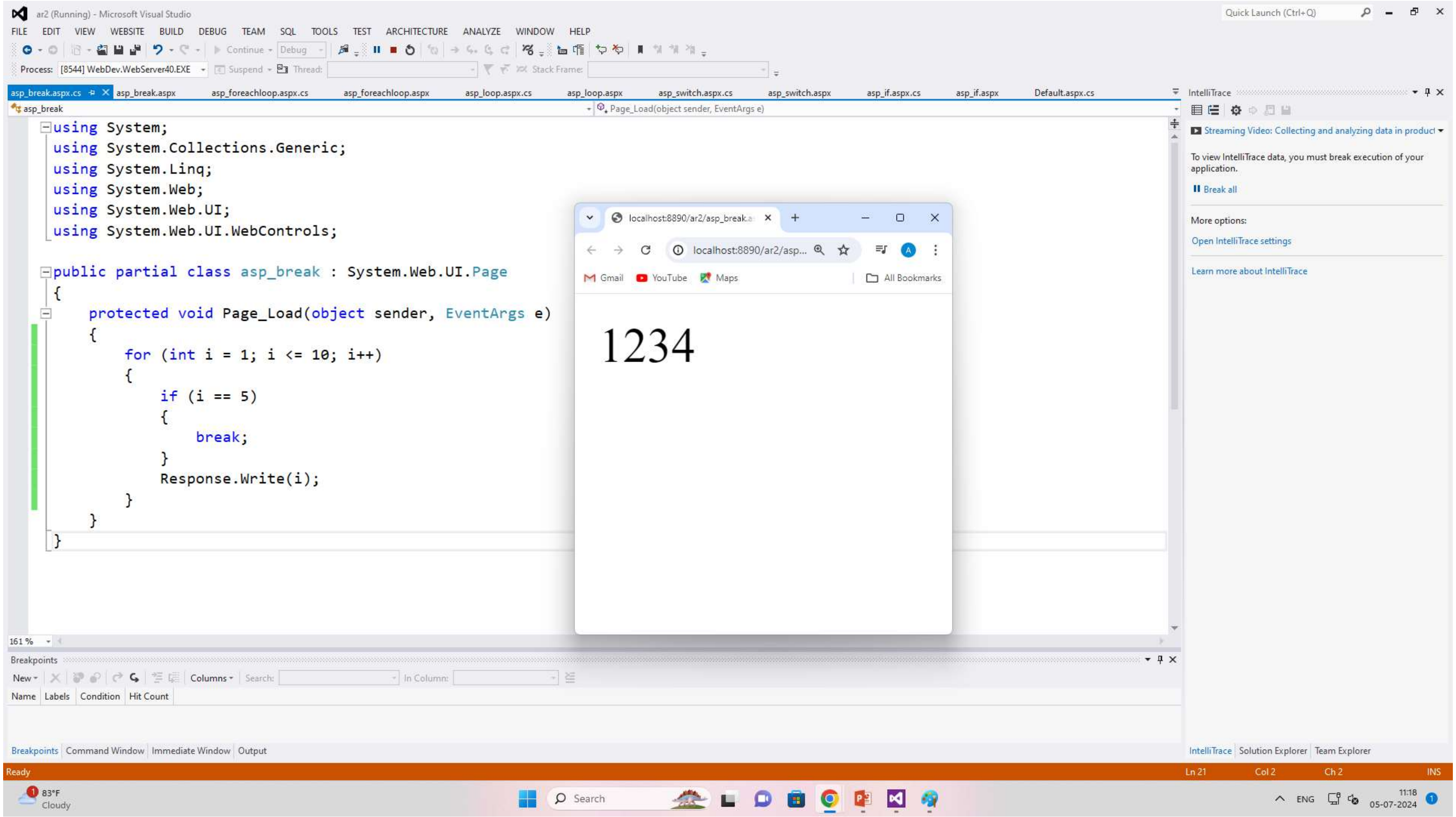
C# (C-Sharp) Jump Statements - Break Statement

✓ The C# *break* is used to break loop or switch statement. It breaks the current flow of the program at the given condition. In case of inner loop, it breaks only inner loop.



Syntax :
jump-statement;
break;

C# (C-Sharp) Jump Statements - Break Statement

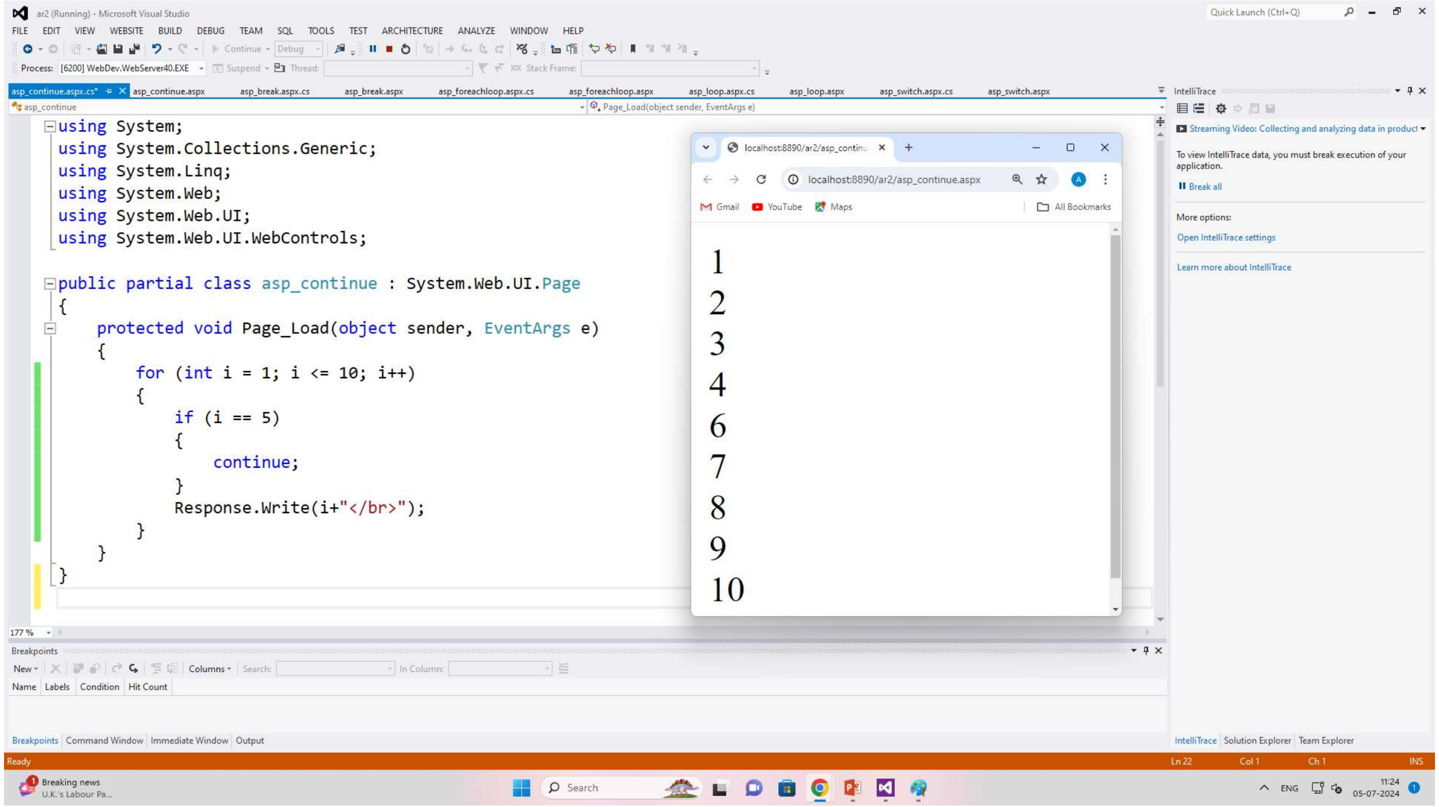


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class asp_break : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        for (int i = 1; i <= 10; i++)
        {
            if (i == 5)
            {
                break;
            }
            Response.Write(i);
        }
    }
}
```

```
1234
```

C# (C-Sharp) Jump Statements –
Continue Statement

- ✓ The C# *continue statement* is used to continue loop. It continues the current flow of the program and skips the remaining code at specified condition. In case of inner loop, it continues only inner loop.


ar2 (Running) - Microsoft Visual Studio
FILE EDIT VIEW WEBSITE BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
asp_continue.aspx.cs
asp_continue
Page_Load(object sender, EventArgs e)
localhost:8890/ar2/asp_continue.aspx
IntelliTrace
Breakpoints

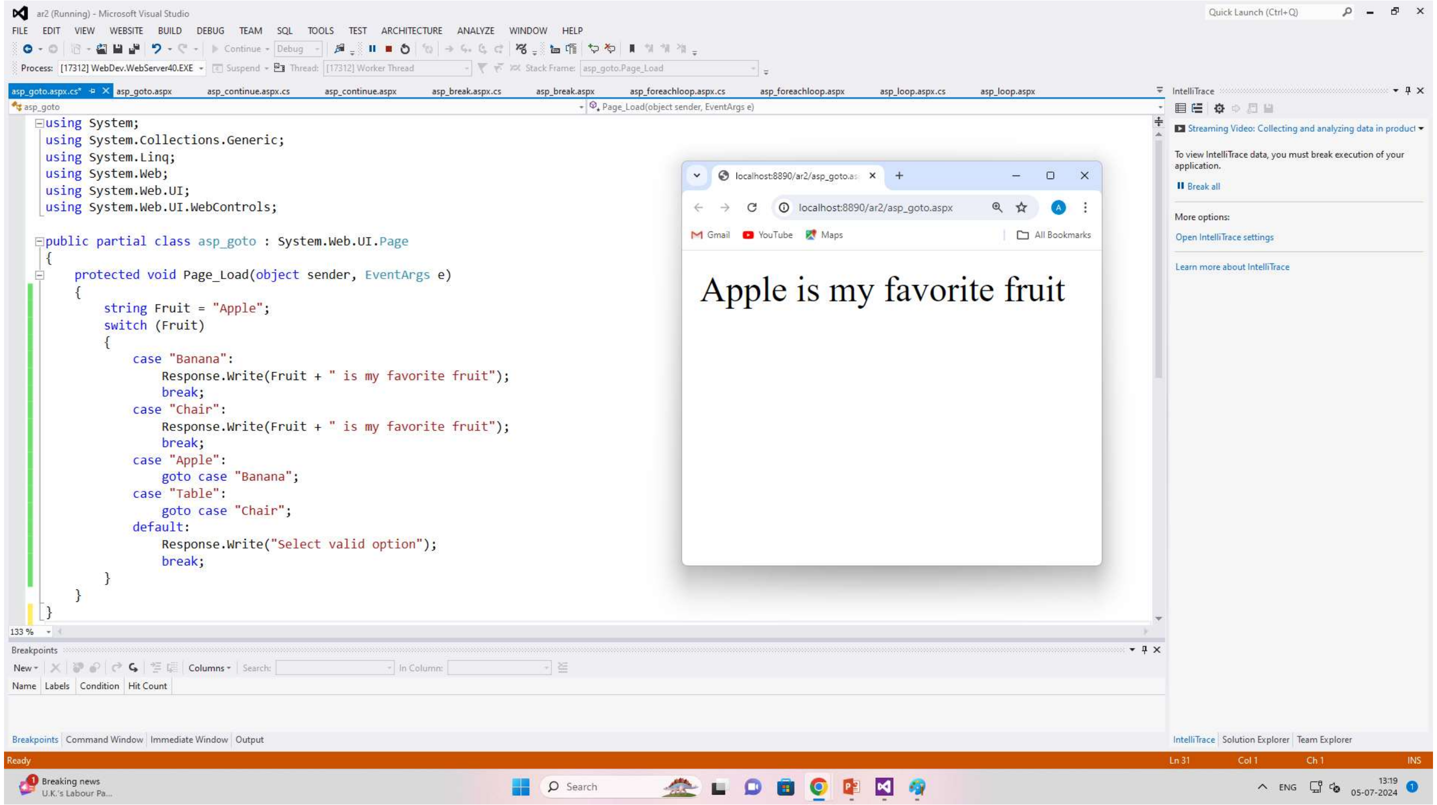
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class asp_continue : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        for (int i = 1; i <= 10; i++)
        {
            if (i == 5)
            {
                continue;
            }
            Response.Write(i+"</br>");
        }
    }
}
```

```
1
2
3
4
6
7
8
9
10
```

Syntax :
jump-statement;
continue;

C# (C-Sharp) Jump Statements – Goto Statement

- ✓ The C# goto statement is also known jump statement. It is used to transfer control to the other part of the program. It unconditionally jumps to the specified label.
- ✓ It can be used to transfer control from deeply nested loop or switch case label.
- ✓ Currently, it is avoided to use goto statement in C# because it makes the program complex.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class asp_goto : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string Fruit = "Apple";
        switch (Fruit)
        {
            case "Banana":
                Response.Write(Fruit + " is my favorite fruit");
                break;
            case "Chair":
                Response.Write(Fruit + " is my favorite fruit");
                break;
            case "Apple":
                goto case "Banana";
            case "Table":
                goto case "Chair";
            default:
                Response.Write("Select valid option");
                break;
        }
    }
}
```

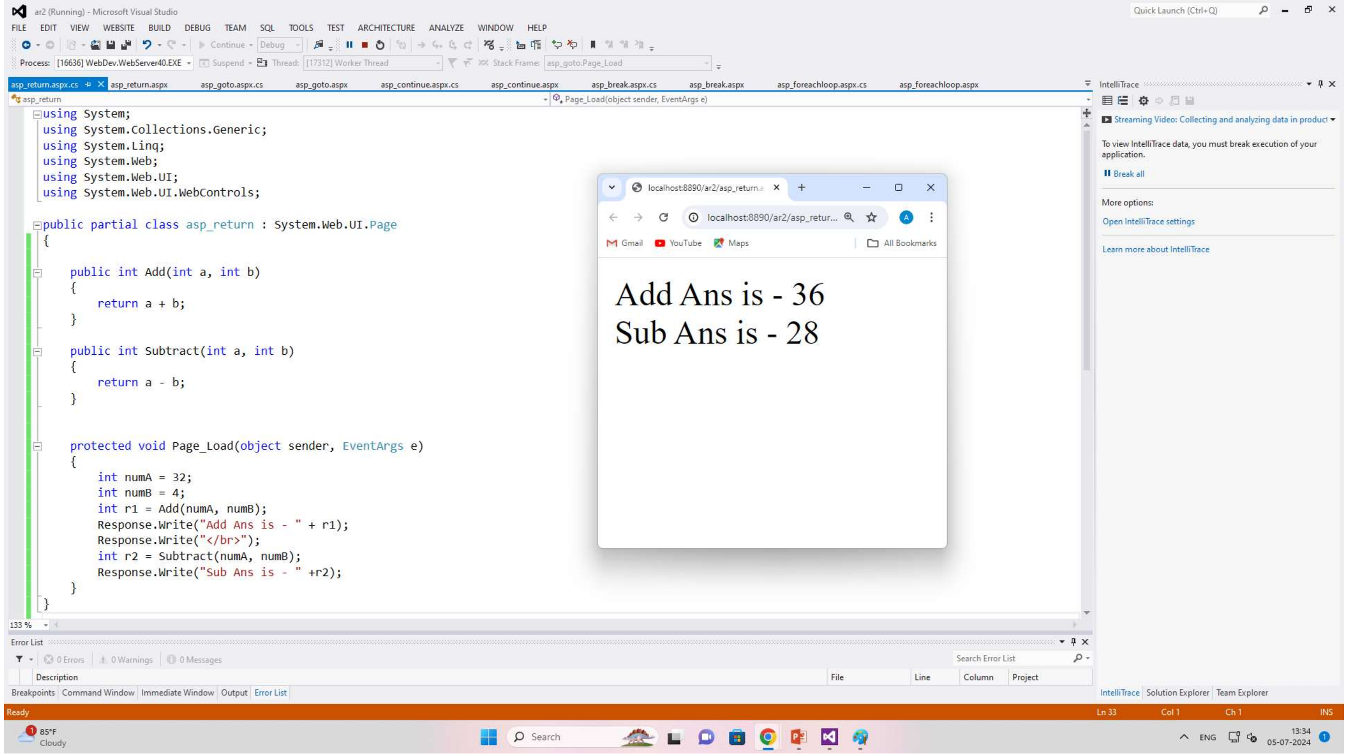
```
Apple is my favorite fruit
```

ar2 (Running) - Microsoft Visual Studio
FILE EDIT VIEW WEBSITE BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Process: [17312] WebDev.WebServer40.EXE
Thread: [17312] Worker Thread
Stack Frame: asp_goto.Page_Load
asp_goto.aspx.cs* asp_goto.aspx asp_continue.aspx.cs asp_continue.aspx asp_break.aspx.cs asp_break.aspx asp_foreachloop.aspx.cs asp_foreachloop.aspx asp_loop.aspx.cs asp_loop.aspx
localhost:8890/ar2/asp_goto.aspx
IntelliTrace
Breakpoints Command Window Immediate Window Output
Ready Ln 31 Col 1 Ch 1 INS

Syntax :
jump-statement;
continue;

C# (C-Sharp) Jump Statements – Return Statement

- ✓ The return statement terminates the execution of the method in which it appears and returns control to the calling method.
- ✓ When the method is executed and returns a value, we can imagine that C# puts this value where the method has been called.
- ✓ The returned value can be used for any purpose from the calling method.
- ✓ Generally, the return statement is useful when you want to get some value from any other method, if there is no need to get value from a method you can use void as a return type.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class asp_return : System.Web.UI.Page
{

    public int Add(int a, int b)
    {
        return a + b;
    }

    public int Subtract(int a, int b)
    {
        return a - b;
    }


    protected void Page_Load(object sender, EventArgs e)
    {
        int numA = 32;
        int numB = 4;
        int r1 = Add(numA, numB);
        Response.Write("Add Ans is - " + r1);
        Response.Write("</br>");
        int r2 = Subtract(numA, numB);
        Response.Write("Sub Ans is - " +r2);
    }
}
```

```
Add Ans is - 36
Sub Ans is - 28
```

C# (C-Sharp) Value Type and Reference Type

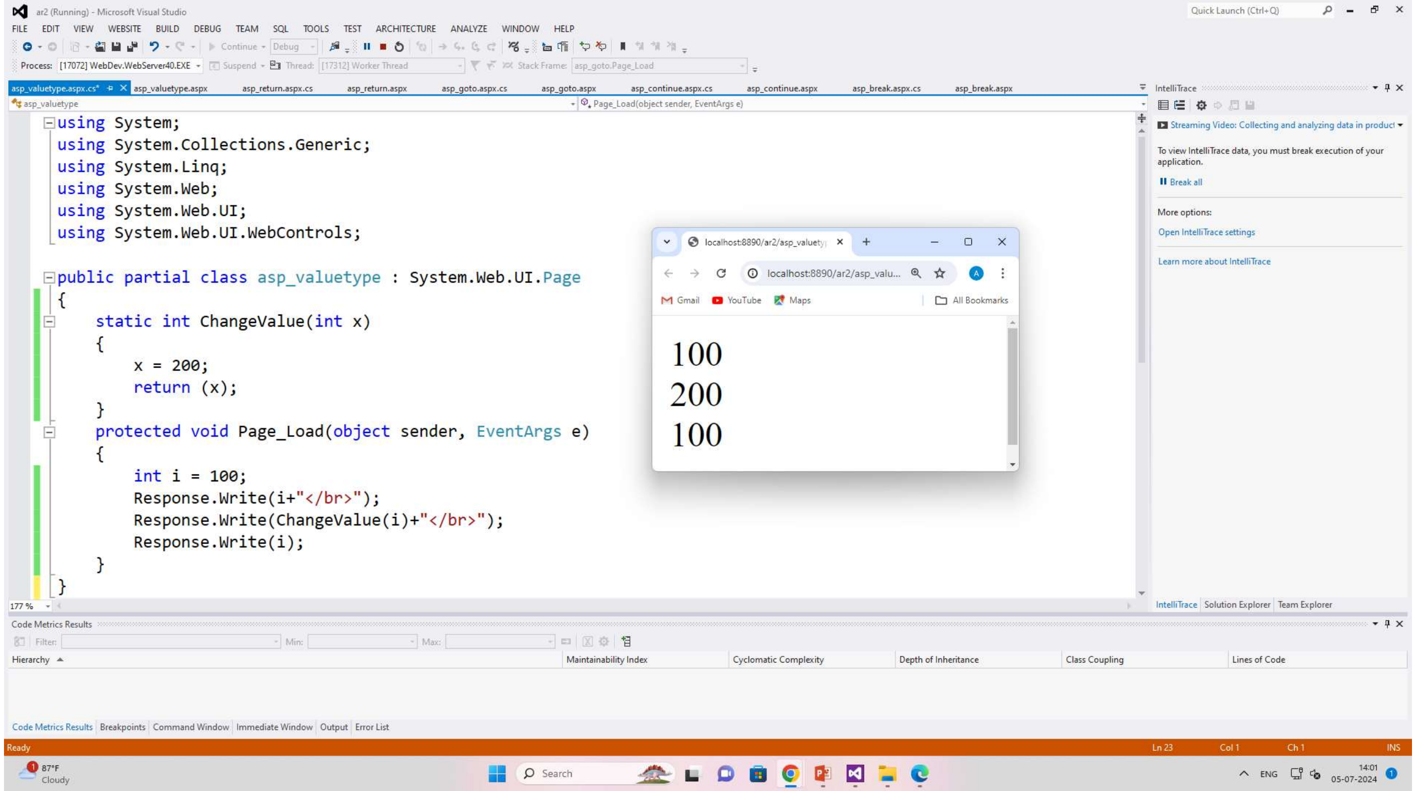
✓ In C#, these data types are categorized based on how they store their value in the memory. C# includes the following categories of data types:

1. Value type 2. Reference type

Value Type

✓ A data type is a value type if it holds a data value within its own memory space. It means the variables of these data types directly contain values.

✓ The following data types are all of value type: bool, byte, char, decimal, double, enum, float, int, long, sbyte, short, struct, uint, ulong, ushort


ar2 (Running) - Microsoft Visual Studio
FILE EDIT VIEW WEBSITE BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Process: [17072] WebDev.WebServer40.EXE
asp_valuetype.aspx.cs* asp_valuetype.aspx asp_return.aspx.cs asp_return.aspx asp_goto.aspx.cs asp_goto.aspx asp_continue.aspx.cs asp_continue.aspx asp_break.aspx.cs asp_break.aspx
Page_Load(object sender, EventArgs e)
IntelliTrace
localhost:8890/ar2/asp_valuetype
Code Metrics Results
Ready

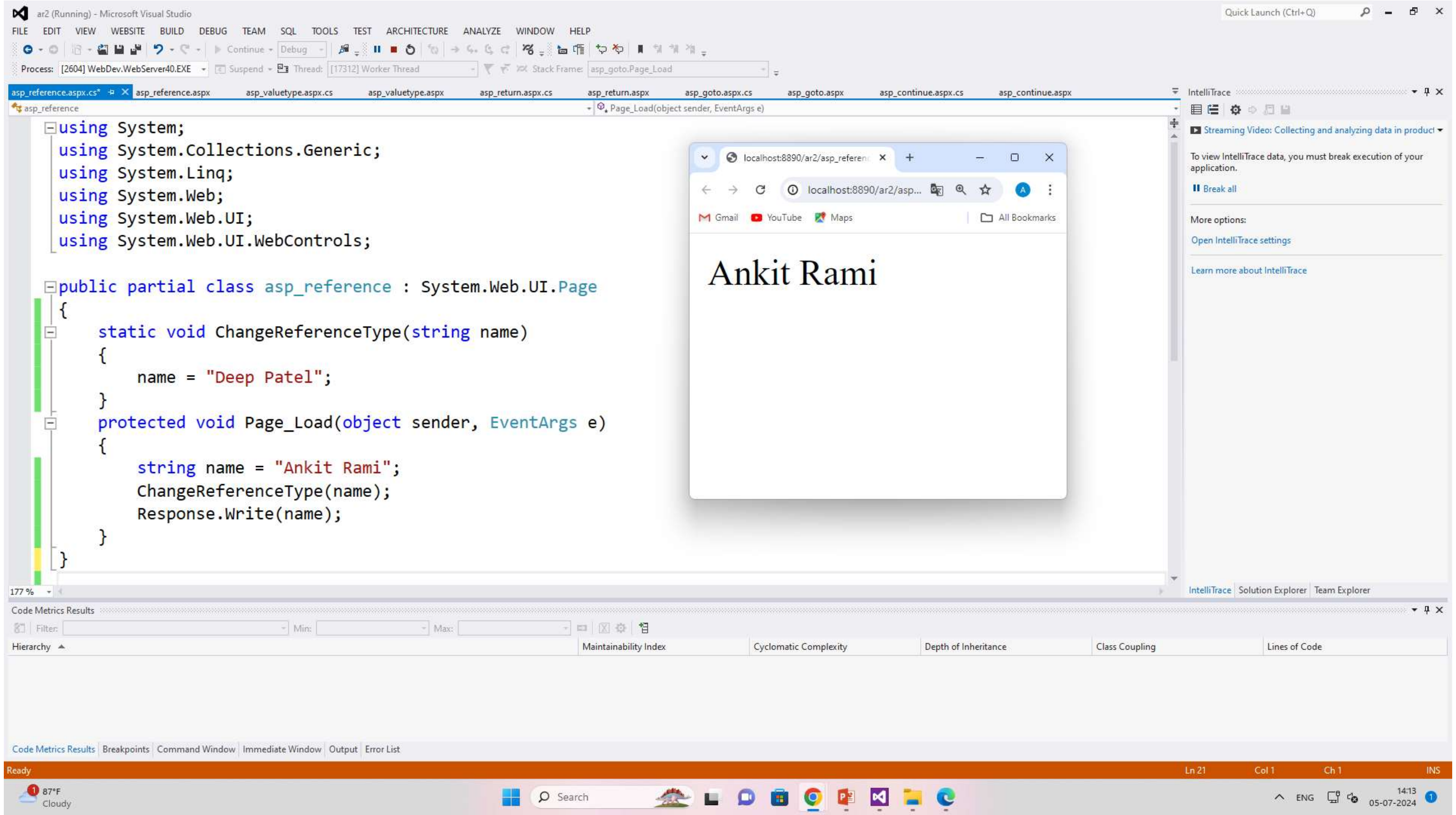
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class asp_valuetype : System.Web.UI.Page
{
    static int ChangeValue(int x)
    {
        x = 200;
        return (x);
    }
    protected void Page_Load(object sender, EventArgs e)
    {
        int i = 100;
        Response.Write(i+"</br>");
        Response.Write(ChangeValue(i)+"</br>");
        Response.Write(i);
    }
}
```

```
100
200
100
```

C# (C-Sharp) Value Type and Reference Type

Reference type

- ✓ Unlike value types, a reference type doesn't store its value directly. Instead, it stores the address where the value is being stored. In other words, a reference type contains a pointer to another memory location that holds the data.
- ✓ The followings are reference type data types: String, Arrays(even if their elements are value types), Class

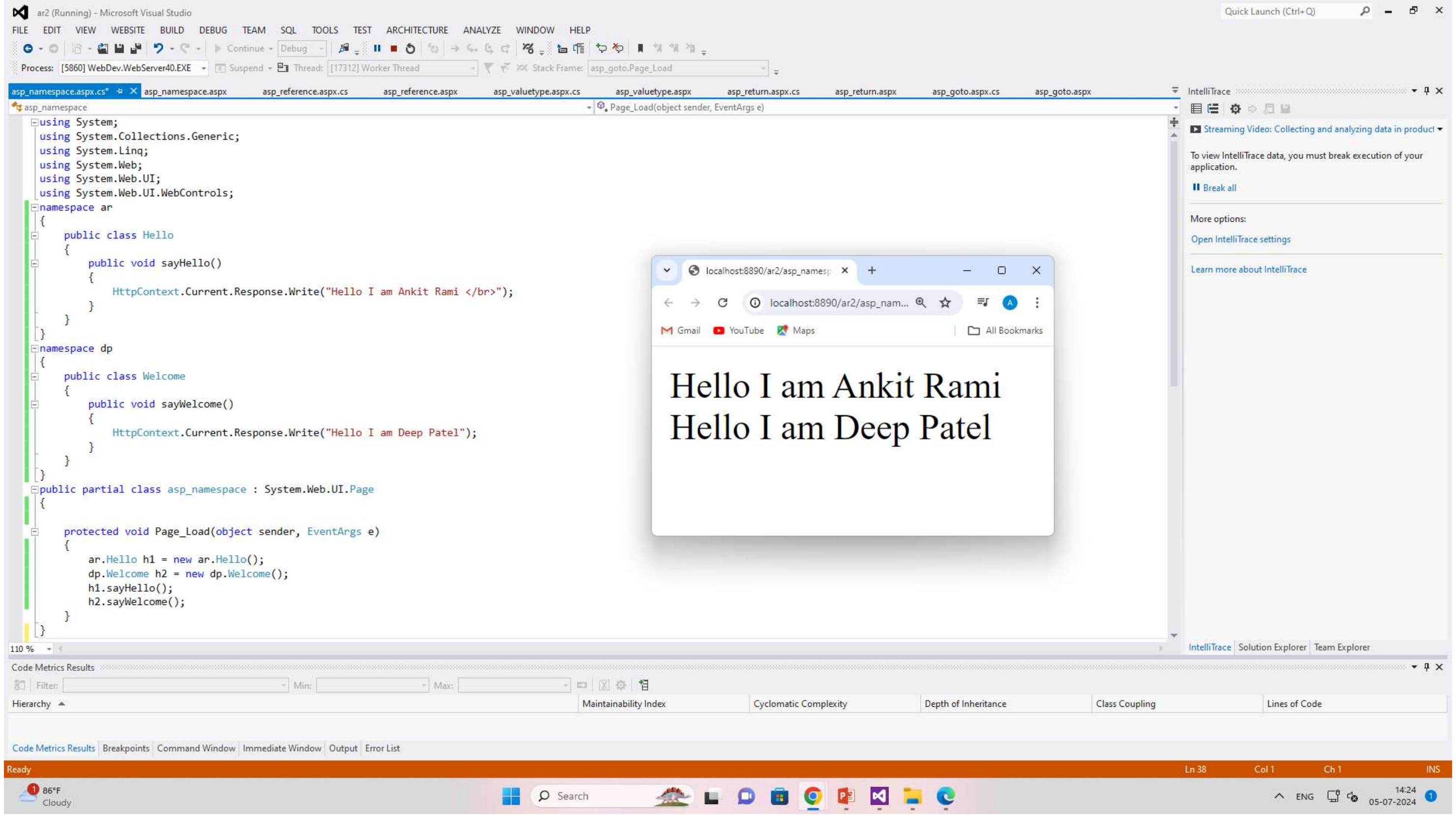

ar2 (Running) - Microsoft Visual Studio
FILE EDIT VIEW WEBSITE BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Process: [2604] WebDev.WebServer40.EXE
Thread: [17312] Worker Thread
Stack Frame: asp_goto.Page_Load
asp_reference.aspx.cs*
asp_reference.aspx
asp_valuetype.aspx.cs
asp_valuetype.aspx
asp_return.aspx.cs
asp_return.aspx
asp_goto.aspx.cs
asp_goto.aspx
asp_continue.aspx.cs
asp_continue.aspx
asp_reference
Page_Load(object sender, EventArgs e)
localhost:8890/ar2/asp_referen
localhost:8890/ar2/asp...
IntelliTrace
Code Metrics Results
Ready

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class asp_reference : System.Web.UI.Page
{
    static void ChangeReferenceType(string name)
    {
        name = "Deep Patel";
    }
    protected void Page_Load(object sender, EventArgs e)
    {
        string name = "Ankit Rami";
        ChangeReferenceType(name);
        Response.Write(name);
    }
}
```

```
Ankit Rami
```

C# (C-Sharp) Namespaces

- ✓ Namespaces in C# are used to organize too many classes so that it can be easy to handle the application.
- ✓ In a simple C# program, we use System.Console where System is the namespace and Console is the class. To access the class of a namespace, we need to use namespace name.classname. We can use **using** keyword so that we don't have to use complete name all the time.
- ✓ In C#, global namespace is the root namespace. The global::System will always refer to the namespace "System" of .Net Framework.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace ar
{
    public class Hello
    {
        public void sayHello()
        {
            HttpContext.Current.Response.Write("Hello I am Ankit Rami </br>");
        }
    }
}
namespace dp
{
    public class Welcome
    {
        public void sayWelcome()
        {
            HttpContext.Current.Response.Write("Hello I am Deep Patel");
        }
    }
}
public partial class asp_namespace : System.Web.UI.Page
{

    protected void Page_Load(object sender, EventArgs e)
    {
        ar.Hello h1 = new ar.Hello();
        dp.Welcome h2 = new dp.Welcome();
        h1.sayHello();
        h2.sayWelcome();
    }
}
```

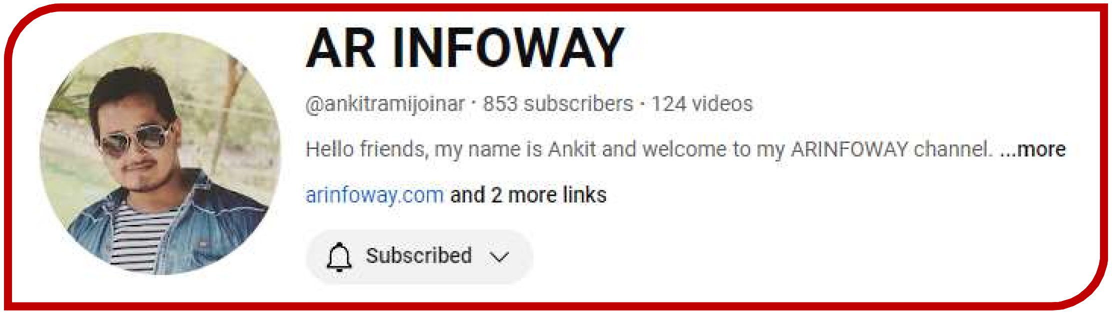
```
Hello I am Ankit Rami
Hello I am Deep Patel
```

ar2 (Running) - Microsoft Visual Studio
FILE EDIT VIEW WEBSITE BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Process: [5860] WebDev.WebServer40.EXE
Thread: [17312] Worker Thread
Stack Frame: asp_goto.Page_Load
asp_namespace.aspx.cs* asp_namespace.aspx asp_reference.aspx.cs asp_reference.aspx asp_valuetype.aspx.cs asp_valuetype.aspx asp_return.aspx.cs asp_return.aspx asp_goto.aspx.cs asp_goto.aspx
Page_Load(object sender, EventArgs e)
localhost:8890/ar2/asp_namesp
Gmail YouTube Maps All Bookmarks
IntelliTrace
To view IntelliTrace data, you must break execution of your application.
Break all
More options:
Open IntelliTrace settings
Learn more about IntelliTrace
Code Metrics Results
Hierarchy Maintainability Index Cyclomatic Complexity Depth of Inheritance Class Coupling Lines of Code
Ready Ln 38 Col 1 Ch 1 INS
86°F Cloudy
14:24 05-07-2024

Reference Link for E-Learning

AMIT ACADEMY
for Computer Education

- https://www.geeksforgeeks.org/client-server-model/
- https://www.simplilearn.com/what-is-client-server-architecture-article
- https://intellipaat.com/blog/what-is-client-server-architecture/
- https://www.perfmatrix.com/software-architecture-and-its-types/
- https://www.javatpoint.com/asp-net-tutorial
- https://www.codekul.com/blog/page-life-cycle-asp-net/
- https://www.tutorialspoint.com/asp.net/asp.net_directives.htm
- https://www.javatpoint.com/c-sharp-tutorial
- https://www.geeksforgeeks.org/csharp-programming-language/?ref=header_search
- https://amit.arinfoway.com

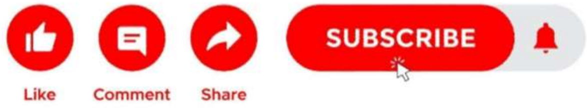


AR INFOWAY

@ankitramijoinar · 853 subscribers · 124 videos

Hello friends, my name is Ankit and welcome to my ARINFOWAY channel. ...more

arinfoway.com and 2 more links



More Information Join YouTube Channel

View More...

Prepared by Ankit Rami(AR) | Any Query Contact – 8460467193 | Email – ankitramiblog@gmail.com