

Introduction to backend development

THE INTRODUCTION TO BACK-END WEB DEVELOPMENT

By Jafar Muzeyin

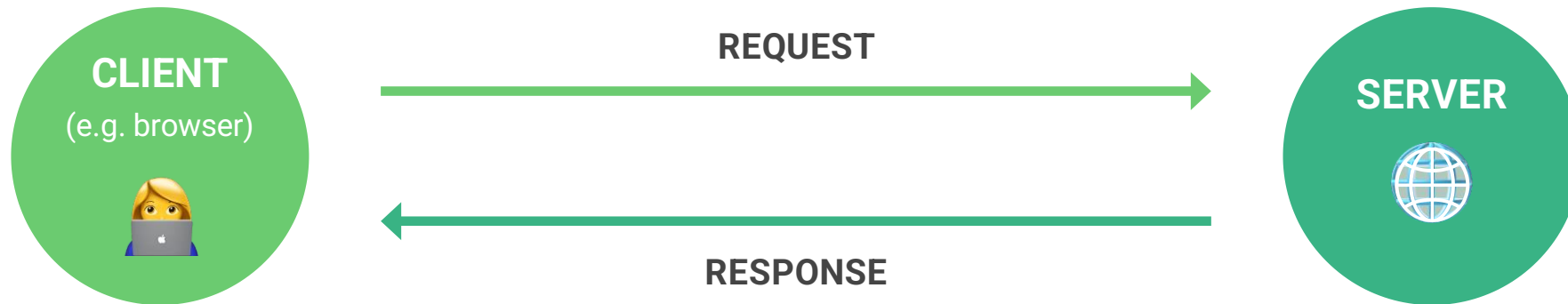
OBJECTIVES

You are going to learn:

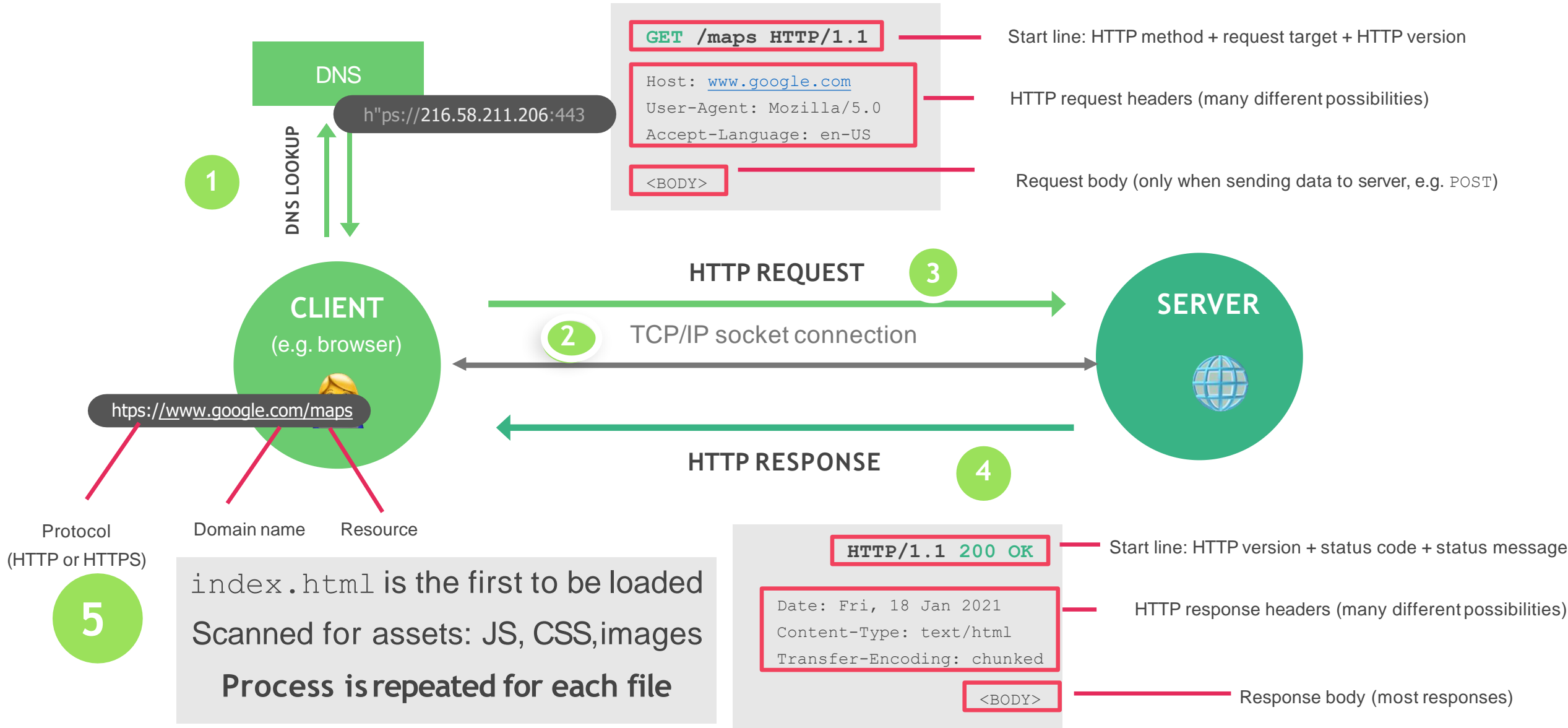
- 👉 The How the web works.
- 👉 Request Response Model
- 👉 Static VS Dynamic Website
- 👉 What API are

WHAT HAPPENS WHEN WE ACCESS WEB PAGE

👉 Request-response model or Client-server architecture



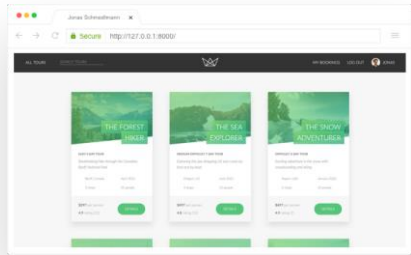
WHAT HAPPENS WHEN WE ACCESS WEB PAGE



FRONT-END AND BACK-END

FRONT-END

BACK-END



BROWSER

WEB SERVER

HTTP
Server

App

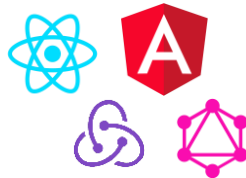
DATABASE

Files

FRONT-END STACK



JS



BACK-END STACK

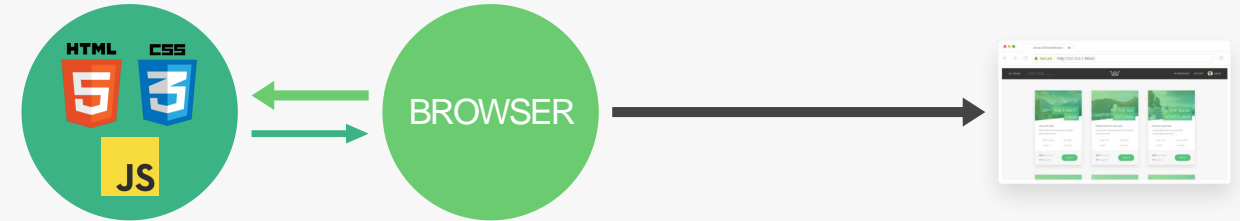
ASP.NET Core

SQL



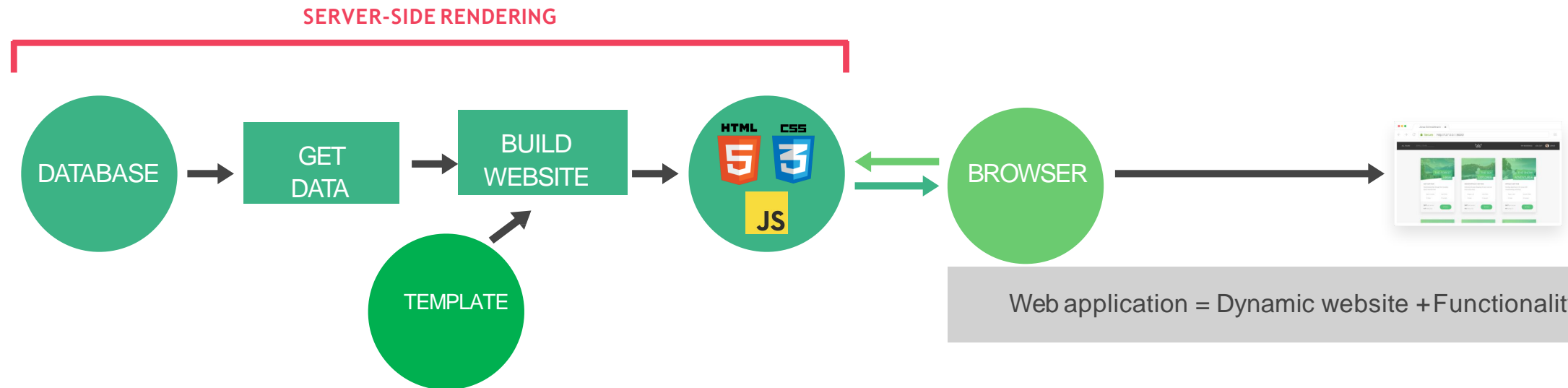
FRONT-END AND BACK-END

STATIC



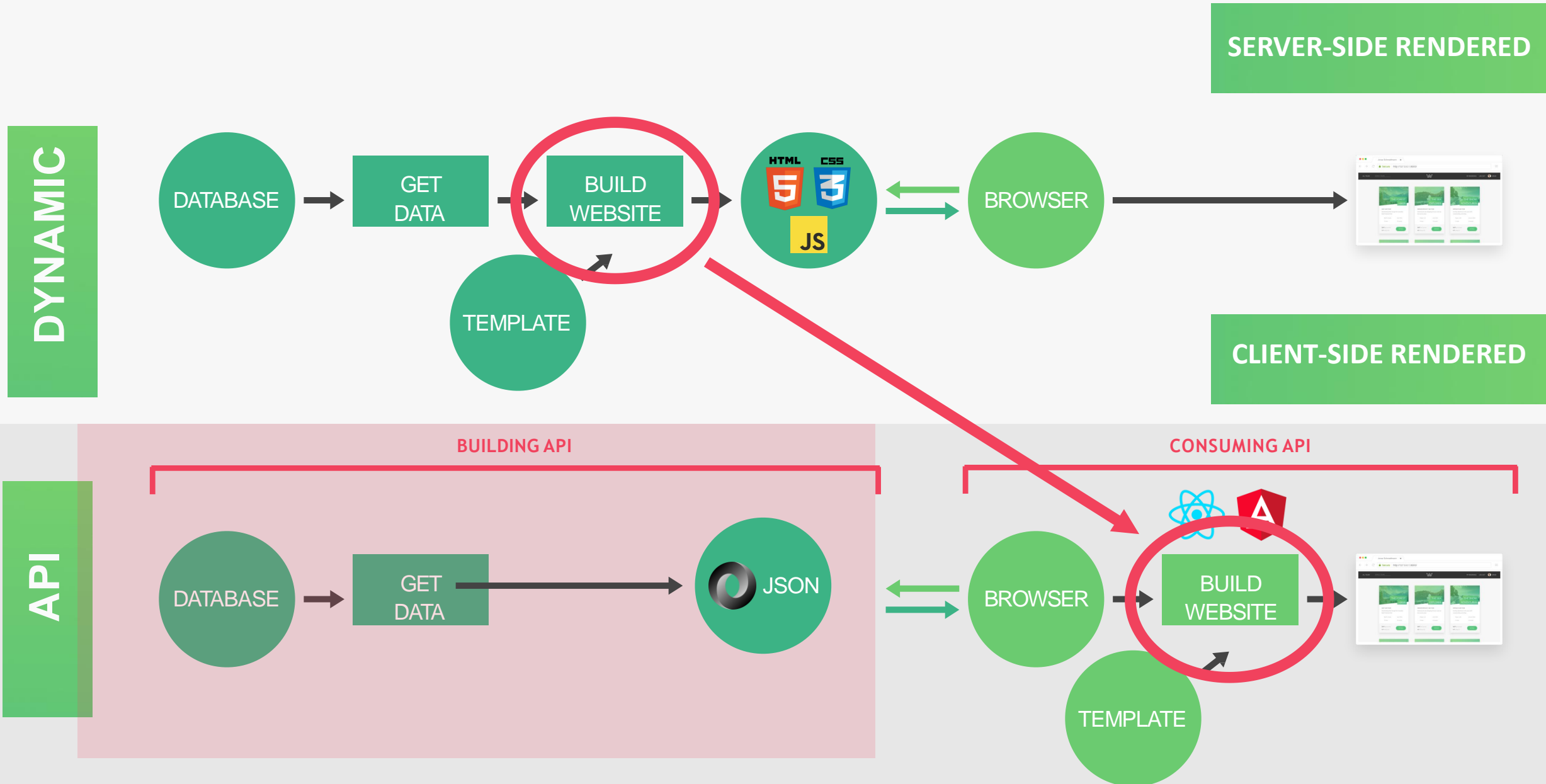
JavaScript \neq Dynamic

DYNAMIC



Web application = Dynamic website + Functionality

DYNAMIC WEBSITES VS API-POWERED WEBSITES

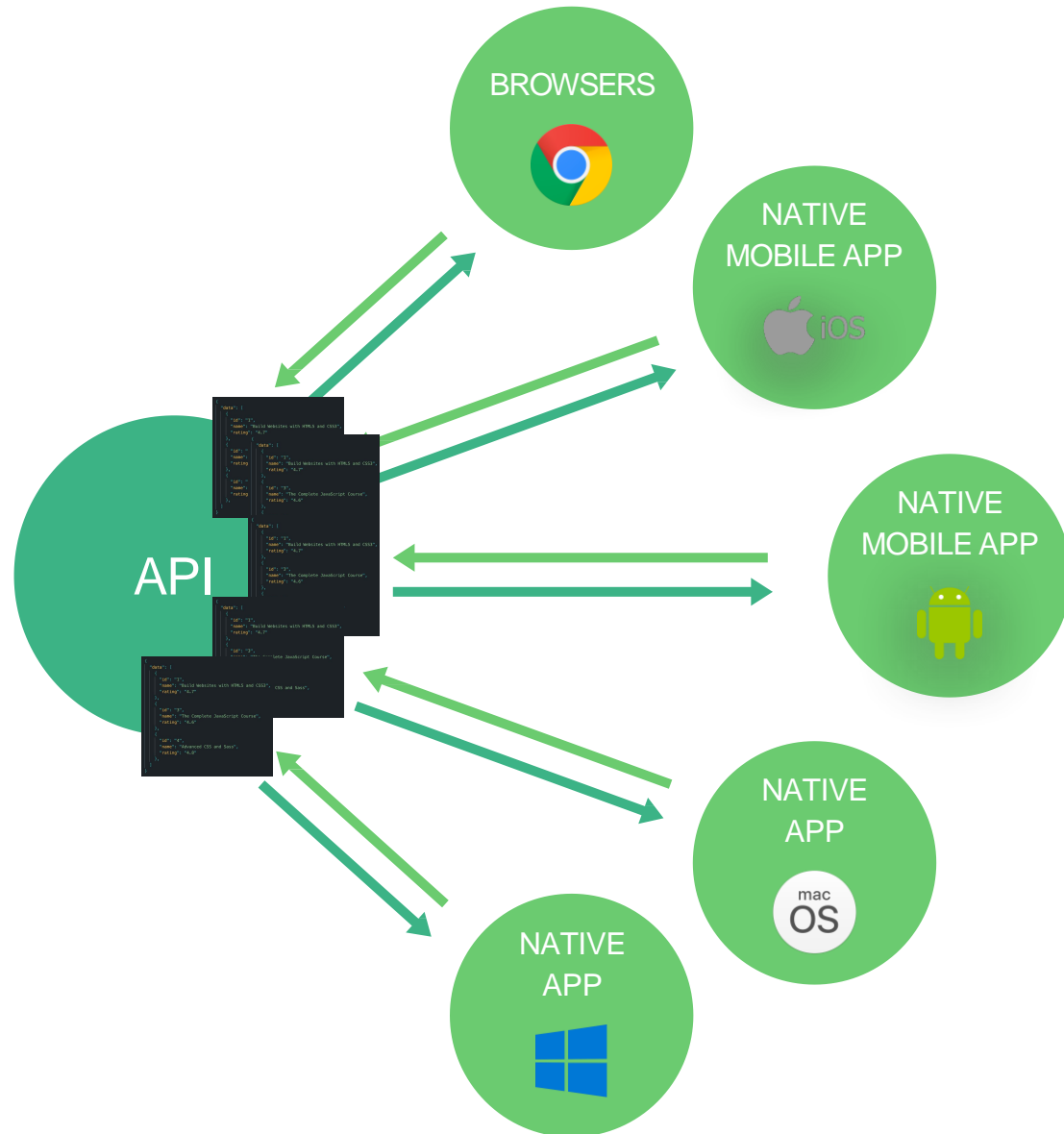


ONE API, MANY CONSUMERS

h"ps://www.hilcoe/api/myCourseData



```
{
  "data": [
    {
      "id": "1",
      "name": "Build Websites with HTML5 and CSS3",
      "rating": "4.7"
    },
    {
      "id": "3",
      "name": "The Complete JavaScript Course",
      "rating": "4.6"
    },
    {
      "id": "4",
      "name": "Advanced CSS and Sass",
      "rating": "4.8"
    }
  ]
}
```







Common Web Application Architecture

Common Web Application Architecture

By Jafar Muzeyin

OBJECTIVES

You are going to learn:

-  Software Architecture
-  Monolithic Architecture
-  Three Layer architecture
-  Microservice architecture

WHY SOFTWARE ARCHITECTURE ?

ARCHITECTURE

*Designing software architecture is about **arranging components** of a system to best fit the desired **quality attributes** of the system.*

QUALITY ATTRIBUTES:

- 👉 **Performance:** how long do you have to wait before that spinning "loading" icon goes away?
- 👉 **Availability:** what percentage of the time is the system running?
- 👉 **Usability:** can the users easily figure out the interface of the system?
- 👉 **Modifiability:** if the developers want to add a feature to the system, is it easy to do?
- 👉 **Scalability:** if you grow your userbase rapidly, can the system easily scale to meet the new traffic?
- 👉 **Deployability:** is it easy to put a new feature in production?
- 👉 **Portability:** can the system run on many different platforms (i.e. Windows vs. Mac vs. Linux)?
- 👉 **Interoperability:** does the system play nicely with other systems?

QUALITY ATTRIBUTES



Performance
Availability
Usability

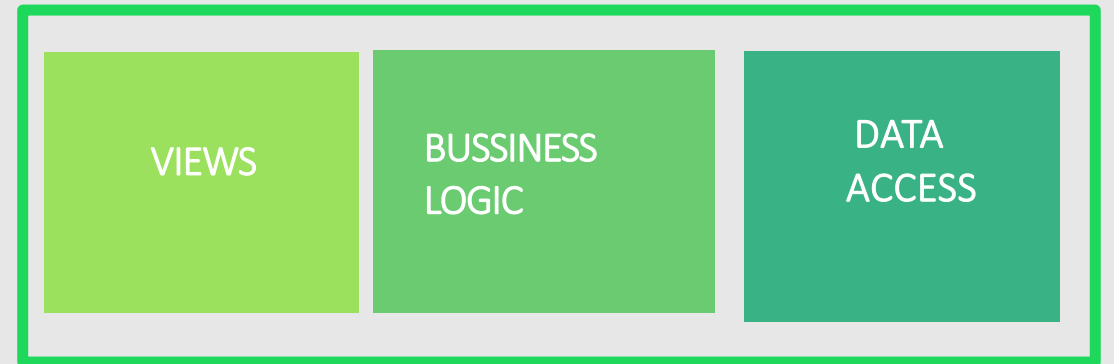


Security
Availability

MONOLOTHIC ARCHITECTURE

- 👉 Single codebase
- 👉 Big deployment
- 👉 Big Single jar/war file

↓
REQUEST



BENEFITS OF MONOLOTHIC ARCHITECTURE

BENEFITS

- 👉 Simple to develop
- 👉 Easier debugging and testing.
- 👉 Very active developer community.

USE

- 👉 To develop simple applications .

CHALLENGES

- 👉 Complicated to understand
- 👉 Making New changes
- 👉 New technology barriers
- 👉 Scalability
- 👉 Difficult to manage
- 👉 Inability to apply new technology

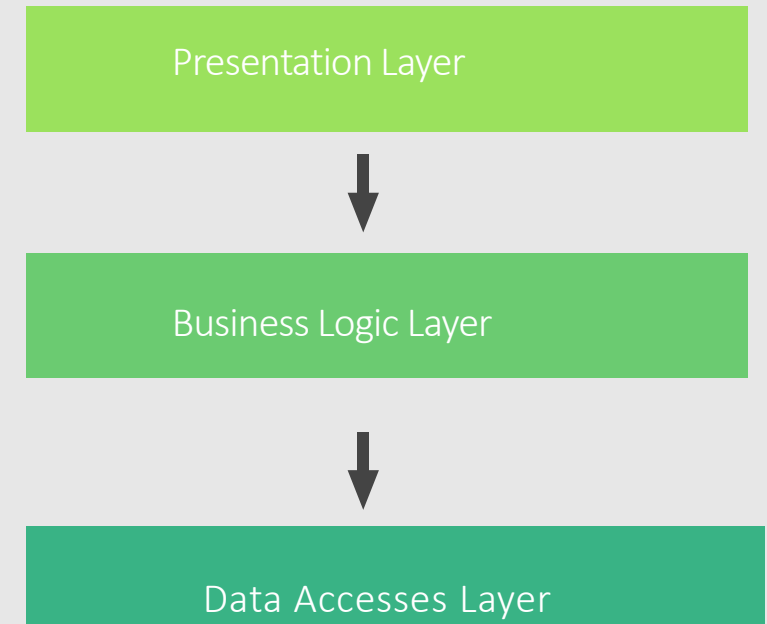
DON'T USE

- 👉 To develop complex applications .

3 Layer Architecture

Three-layer architecture

- Three-layer architecture is a **client-server architecture** in which the functional processes of **user interface**, **business logic** and **data access** layer are developed and maintained as independent modules separate platforms.

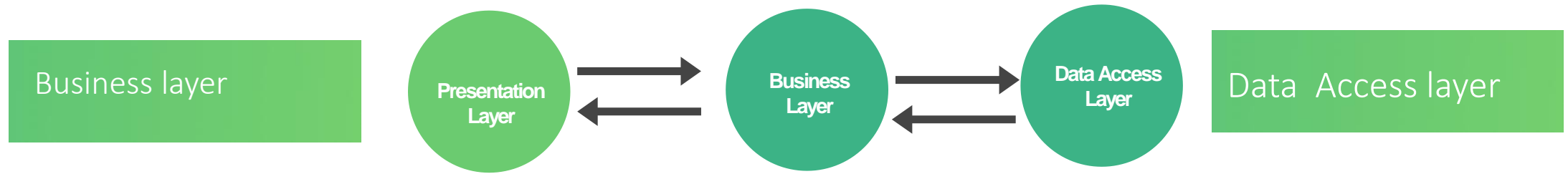


3 Layer Architecture

Presentation layer

- 👉 **The presentation layer** : is used to display the data to the users
- 👉 **.html or .aspx Pages** : are presentation layer which shows data to the users.
- 👉 **Presentation Layer** : is mainly used for getting user data and then passing it to Business Logic Layer for further procedure, and when data is received in Value Object then it's responsible to represent value object in the appropriate form which user can understand.

3 Layer Architecture



- 👉 Business Layer is the middle layer or bridge layer that connects database layer and presentation layer
- 👉 Directly related to **business rules**, how the business works, and business needs.
- 👉 This layer communicates with database layer and presentation layer.

- 👉 Database layer which makes the connection to the database server.
- 👉 In this layer you can write database connection and SQL queries or stored procedures.
- 👉 This layer only communicates with business layer.

Examples

- 👉 Creating new products in the database;
- 👉 Checking if user's password is correct;
- 👉 Validating user input data;
- 👉 Ensuring only users who pay can get product.

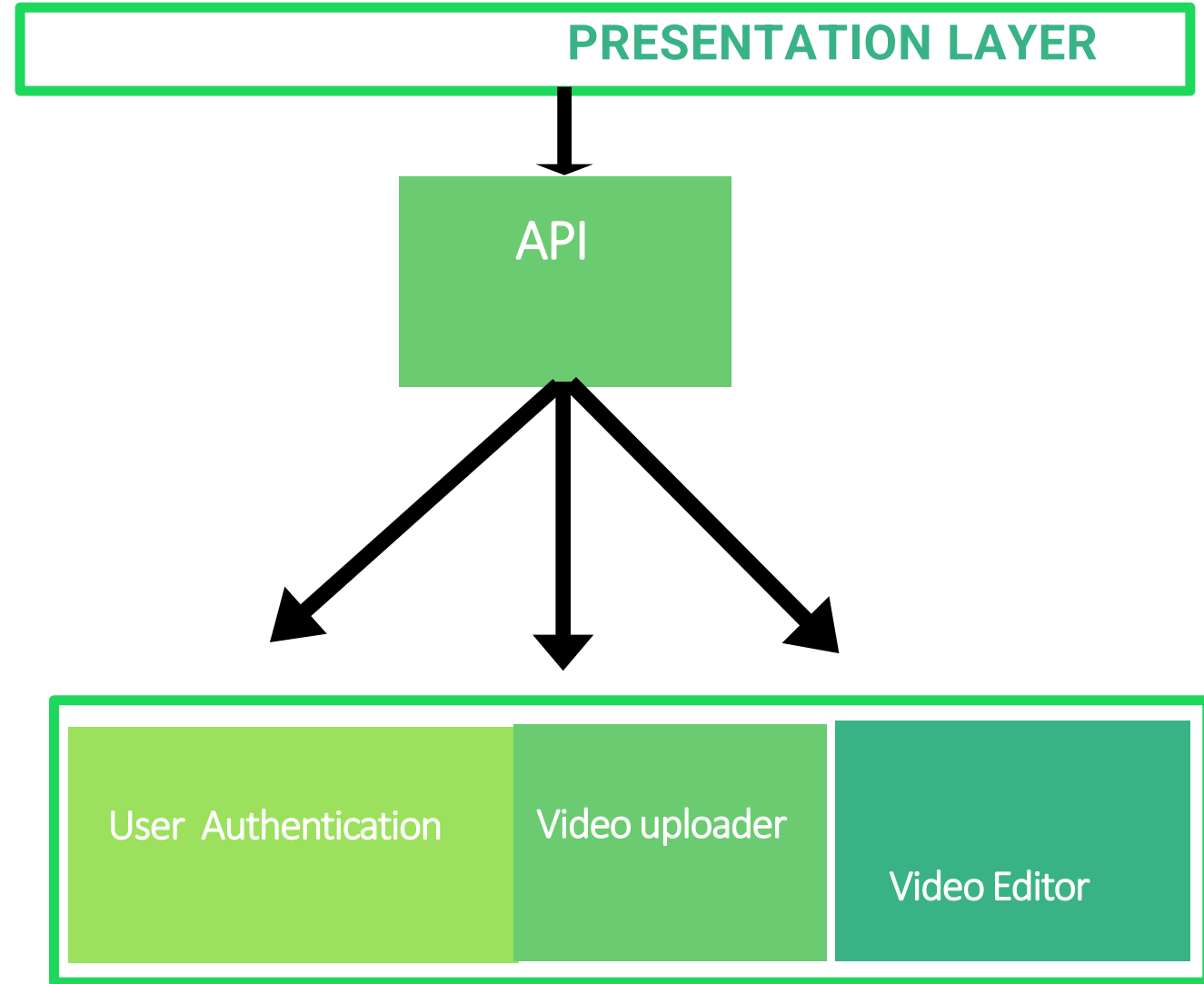
👉 **The three-layer architecture** is a software design pattern and well-established software architecture

What are the advantages of three layers?

- 👉 **Maintainability:** Because each layer is independent of the other layer, updates or changes can be carried out without affecting the application as a whole;
- 👉 **Flexibility:** Because each tier can be managed or scaled independently, flexibility is increased.

MICROSERVICE ARCHITECTURE

- 👉 Running in its own process
- 👉 Communicating with APIs
- 👉 Independently deployable
- 👉 Own technology stack
- 👉 Decouple microservices with bounded context



Three Tier Architecture

OBJECTIVES

You are going to learn:

- 👉 What is Tier.
- 👉 Types of Tier
- 👉 Layer Vs Tier

WHAT IS TIER?

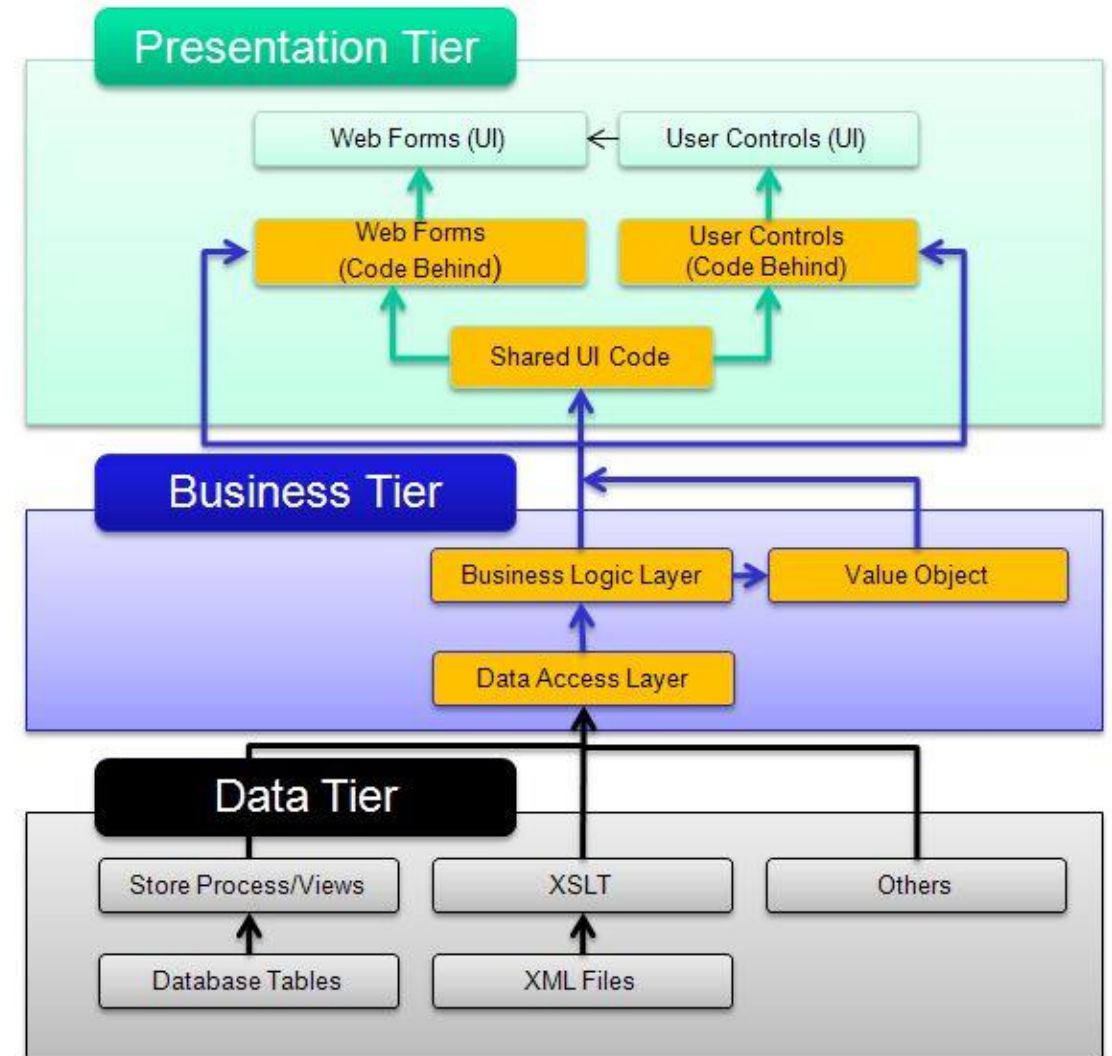
TIER

- 👉 Tier indicates a physical separation of components, which may mean different assemblies such as DLL, EXE, etc. on the same server or multiple servers.

WHAT IS PRESENTATION TIER

PRESENTATION TIER

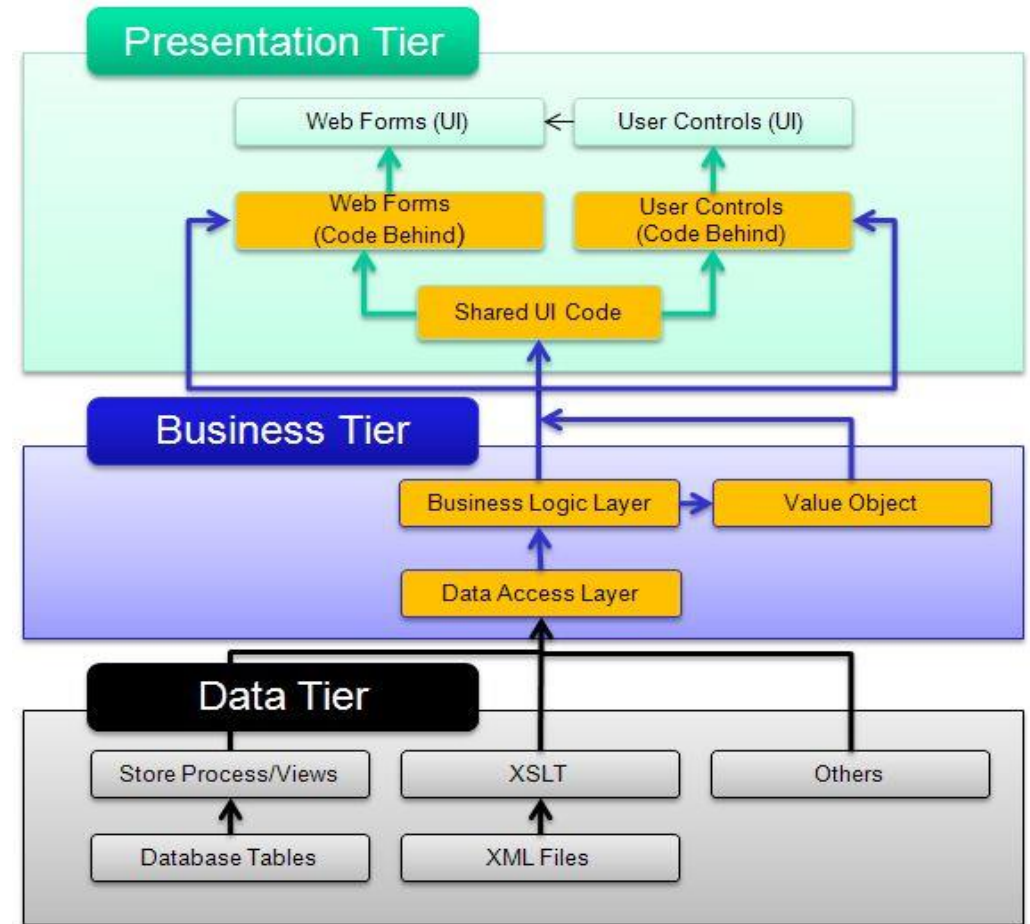
- 👉 Presentation Tier is the tier in which the users interact with an application.
- 👉 Presentation Tier contains Shared UI code, Code Behind and Designers used to represent information to user.



WHAT IS BUSINESS TIER?

BUSINESS TIER

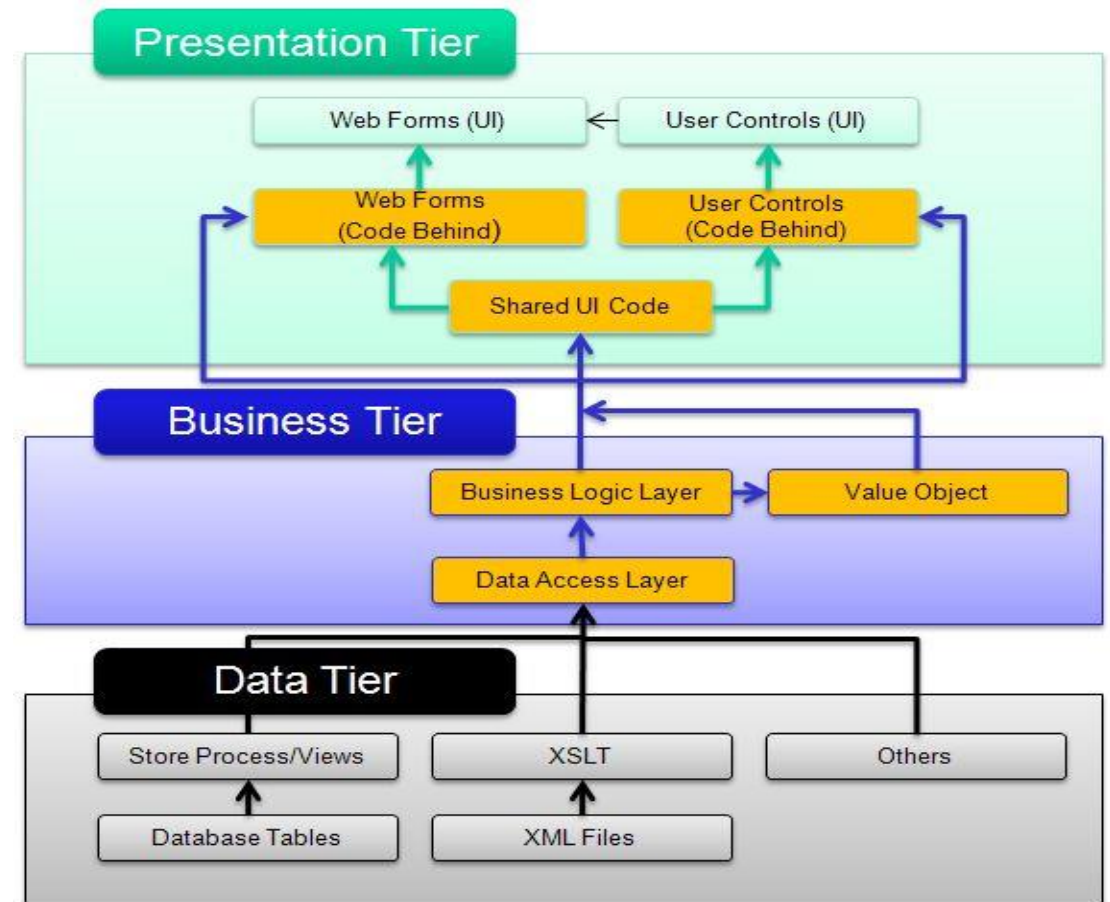
- ☞ Mainly working as the bridge between Data Tier and Presentation Tier.
- ☞ Contents is the sum of Business Logic Layer, Data Access Layer and Value Object and other components used to add business logic.
- ☞ All the Data passes through the Business Tier before passing to the presentation Tier.



WHAT IS DATA TIER ?

DATA TIER

- 👉 Data Tier is basically the server which stores all the application's data.
- 👉 Data tier contents Database Tables, XML Files and other means of storing Application Data.



LAYER VS TIER

LAYER

- 👉 Layer: Layer indicates logical separation of components, such as having distinct namespaces and classes for the Database Access Layer, Business Logic Layer and User Interface Layer
- 👉 A way of organizing a code into a set of layers defined by specific function.

TIER

- 👉 Tiers however, are only about where the code runs.
- 👉 Specifically, tiers are places where layers are deployed and where layers run. In other words, tiers are the physical deployment of layers.

Web Presentation Patterns

WEB PRESENTATION PATTERN

By Jafar Muzeyin

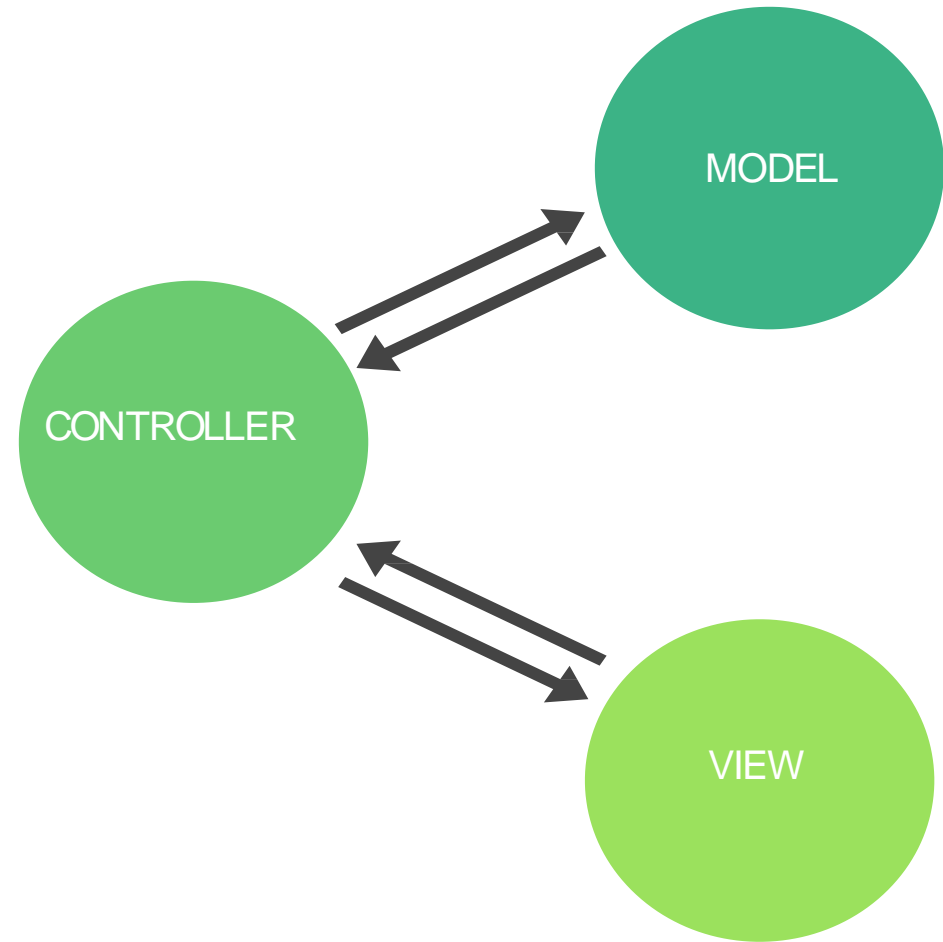
OBJECTIVES

You are going to learn:

- 👉 MVC is a software architecture pattern
- 👉 What is Model
- 👉 What is View
- 👉 What is Control
- 👉 Application vs. Business Logic

MVC ARCHITECTURE

- 👉 MVC is a software architecture pattern which follows the separation of concerns method.
- 👉 On MVC model .NET applications are divided into three interconnected parts which are called Model, View, and Controller.



What is Model?

Model

- Represents business logic and data.
- Contains properties and application logic.
- Communicates with the database, i.e., retrieves data and stores data in the database.

What is View?

View

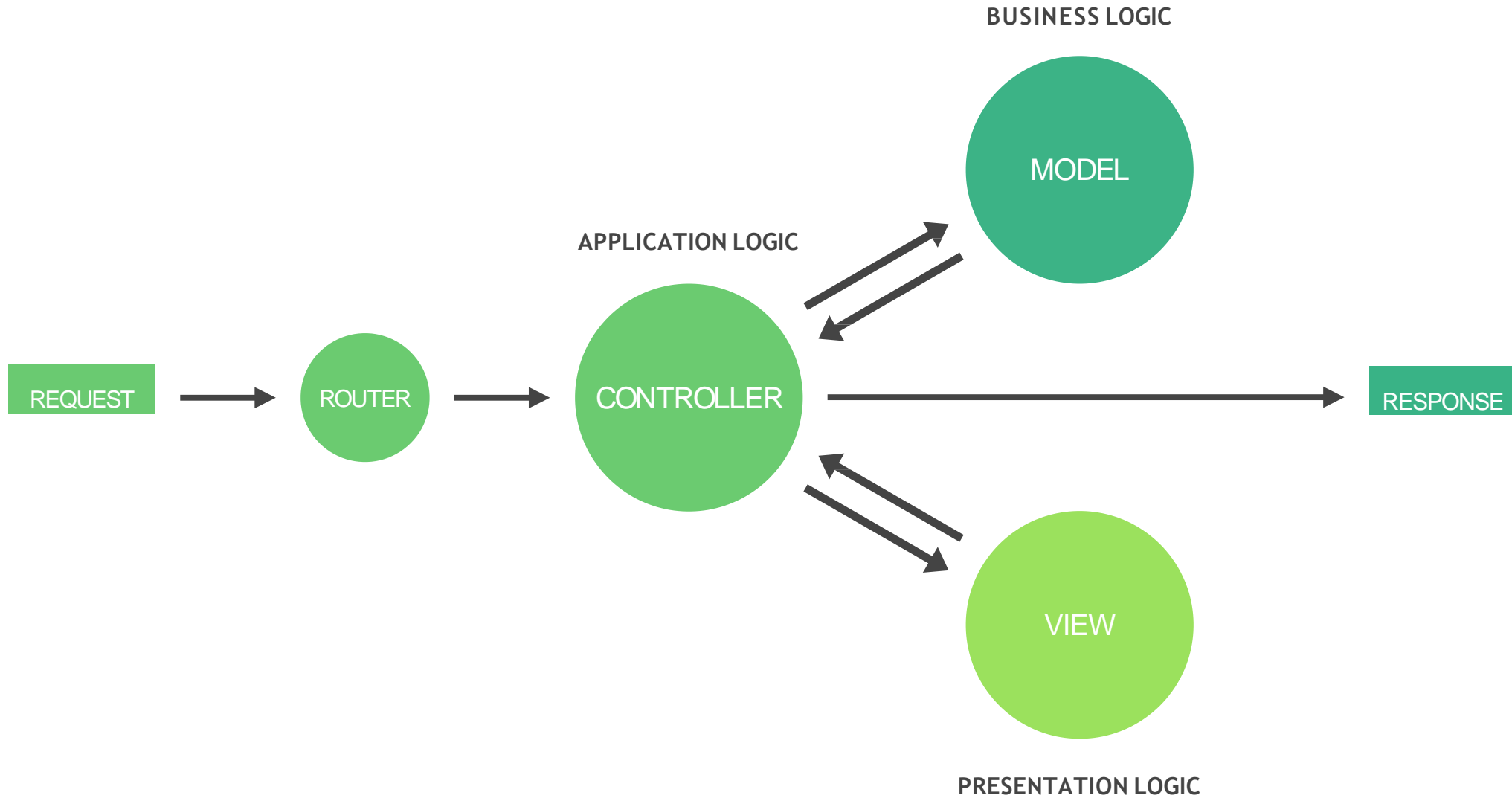
- 👉 Represents the presentation layer of the application.
- 👉 Responsible for providing the User Interface (UI) to the user.
- 👉 There is no input logic or processing of user actions, these are managed inside the controller. This separation makes the application more testable.

What is Controller?

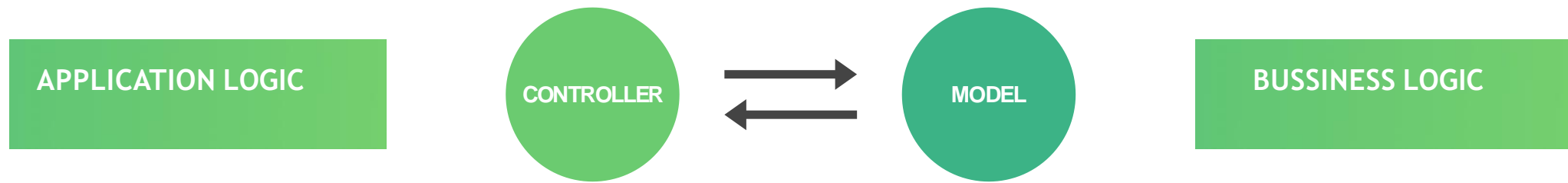
Controller

- 👉 The controller acts as a coordinator between the Model and the View.
- 👉 Manages and responds to user input & interaction.
- 👉 HTTP requests are routed to individual controllers and then the controller calls the model and selects a View for displaying the results..

MVC ARCHITECTURE



APPLICATION LOGIC VS BUSSINESS LOGIC



- 👉 Code that is only concerned about the application's implementation, not the underlying business problem we're trying to solve (e.g. showing and selling products);

- 👉 Concerned about managing requests and responses;

- 👉 About the app's more technical aspects;

- 👉 Bridge between model and view layers.

- 👉 Code that actually solves the business problem we set out to solve;

- 👉 Directly related to business rules, how the business works, and business needs;

- 👉 Examples:

- 👉 Creating new tours in the database;

- 👉 Checking if user's password is correct;

- 👉 Validating user input data;

- 👉 Ensuring only users who bought a tour can review it.

👉 **Fat models/thin controllers:** offload as much logic as possible into the models, and keep the controllers as simple and lean as possible.

Introduction to ASP.NET

ASP.NET

By Jafar Muzeyin

OBJECTIVES

You are going to learn:

- 👉 Defining ASP.NET
- 👉 How ASP.NET is different from ASP
- 👉 CORE STRENGTH OF ASP.NET
- 👉 ASP.NET Platform Frameworks

ASP.NET

INTRODUCTION TO ASP.NET

BY JAFAR MUZEYIN

WHAT IS ASP.NET?

ASP.NET

ASP.NET PROVIDES SERVICES TO ALLOW THE
CREATION, DEPLOYMENT, AND EXECUTION OF
WEB APPLICATIONS AND WEB SERVICES . 🤔

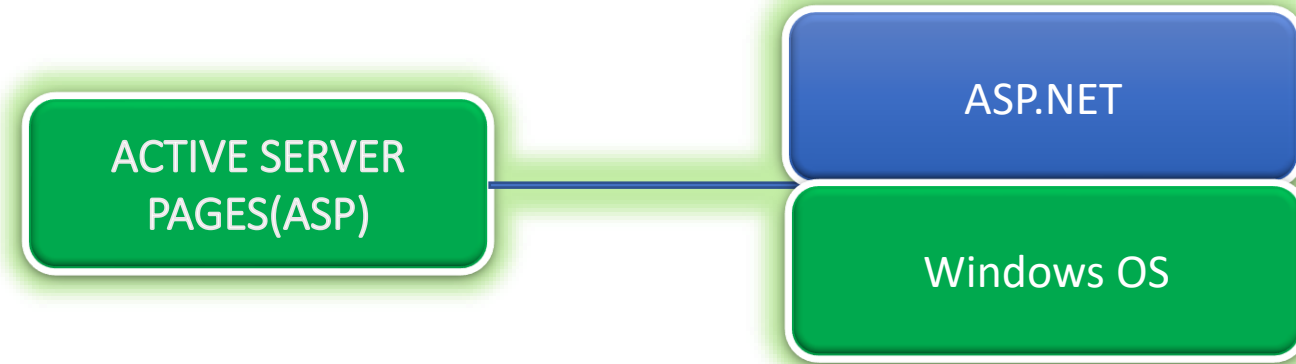
WHAT IS ASP.NET?

ASP.NET

- 👉 ASP.NET is designed first and foremost as a server-side programming platform.
- 👉 That means that all ASP.NET code runs on the web server.

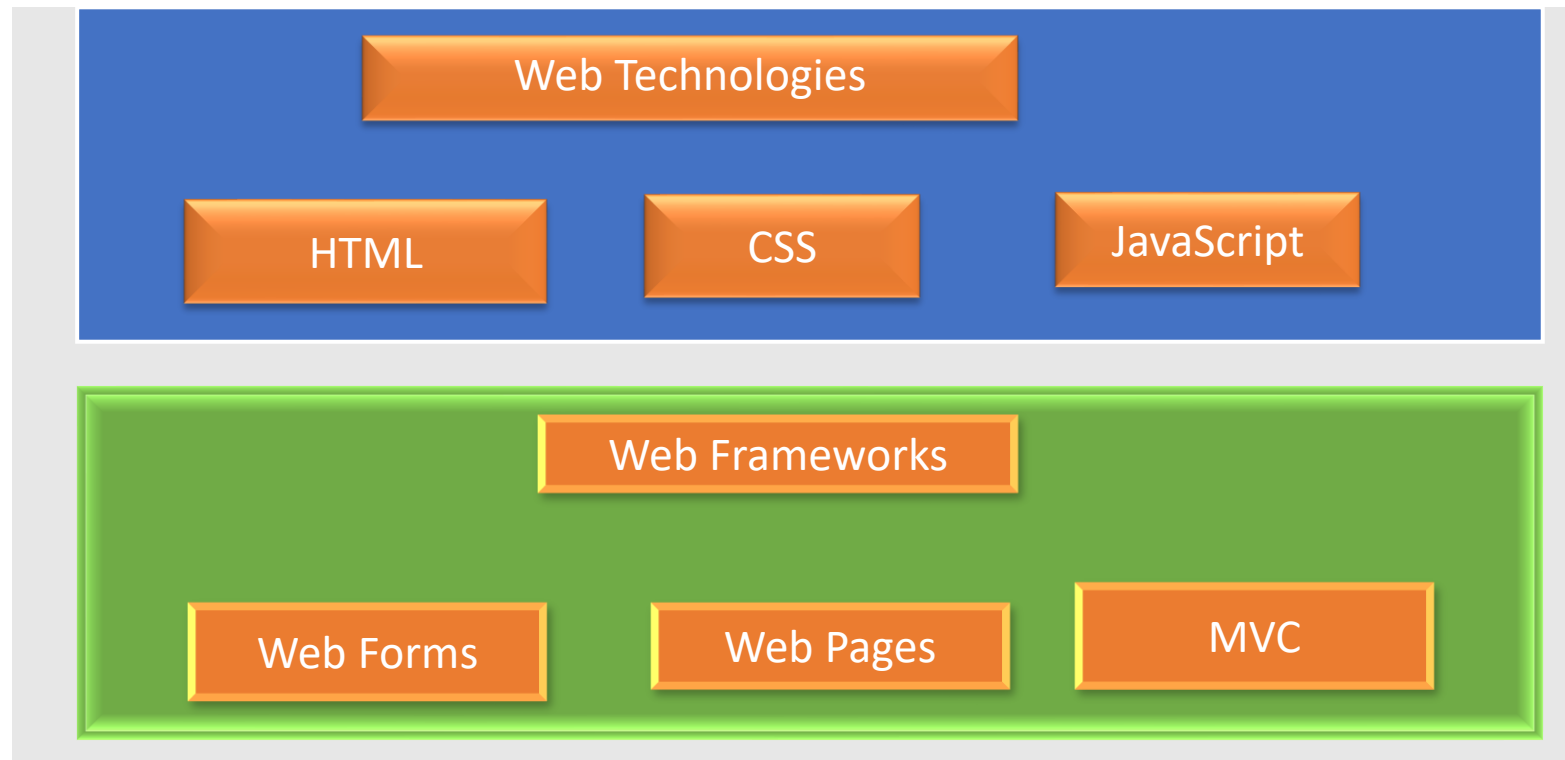
How ASP.NET is different from ASP

- 👉 ASP: server side technology for creating dynamic web pages using scripting language eg: vb script.
- 👉 ASP.NET: server side technology for creating dynamic web pages using Fully Fledged programming languages supported by .NET



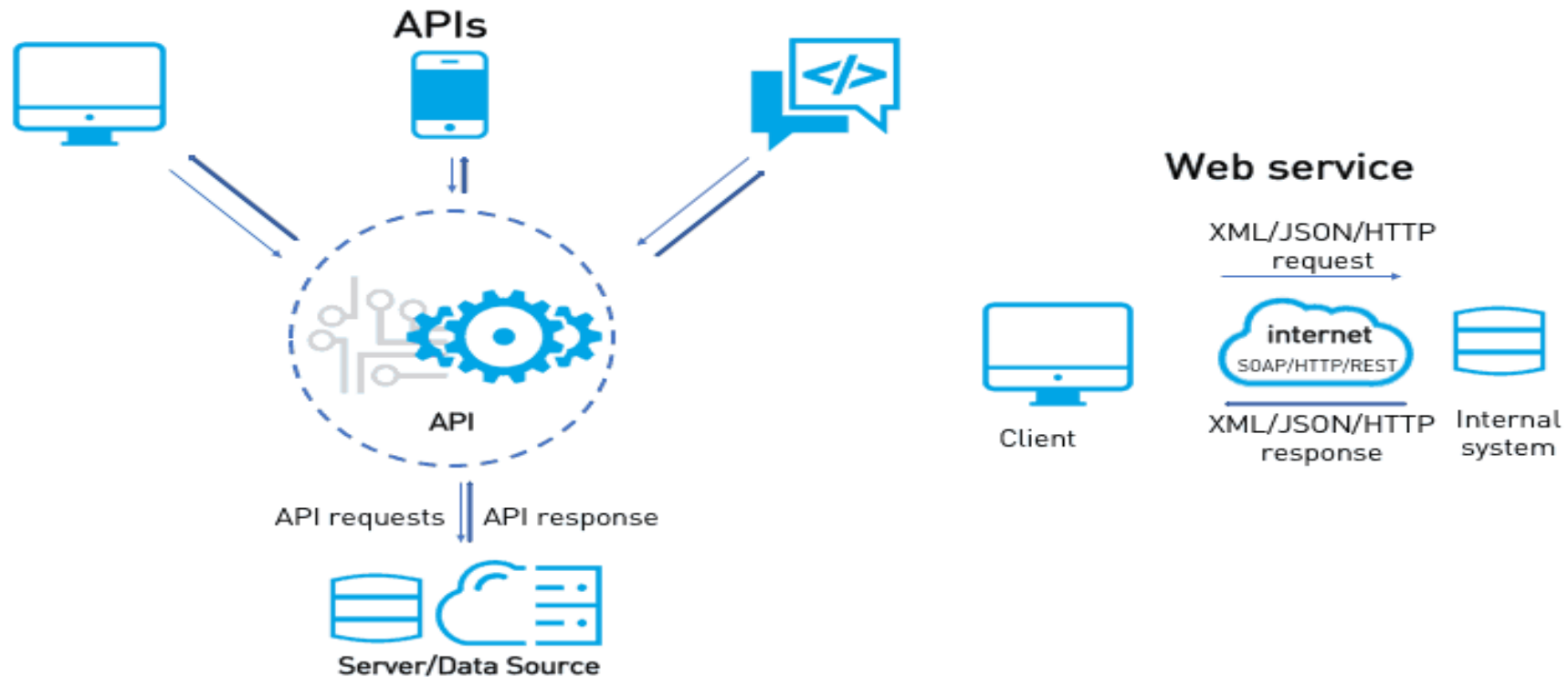
WHAT ARE THE CORE STRENGTHS OF ASP.NET?

To build dynamic website with familiar technologies like HTML, CSS, and JavaScript.



WHAT ARE THE CORE STRENGTHS OF ASP.NET?

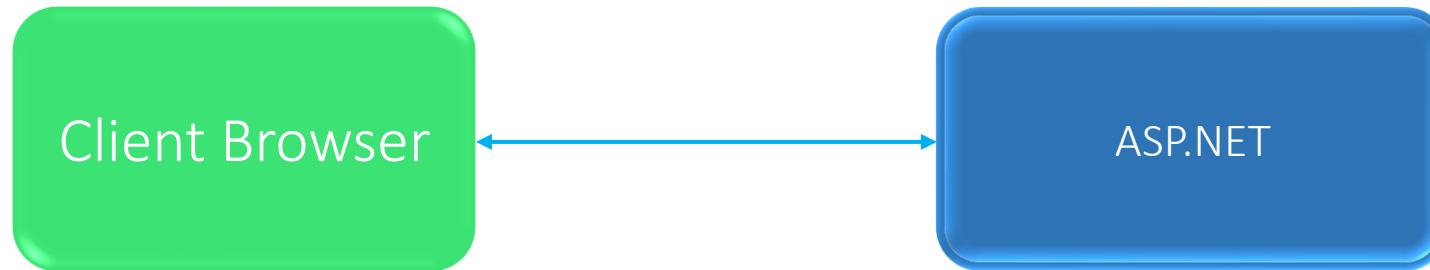
It offers a framework for building RESTful HTTP services called ASP.NET Web API.



WHAT ARE THE CORE STRENGTHS OF ASP.NET?

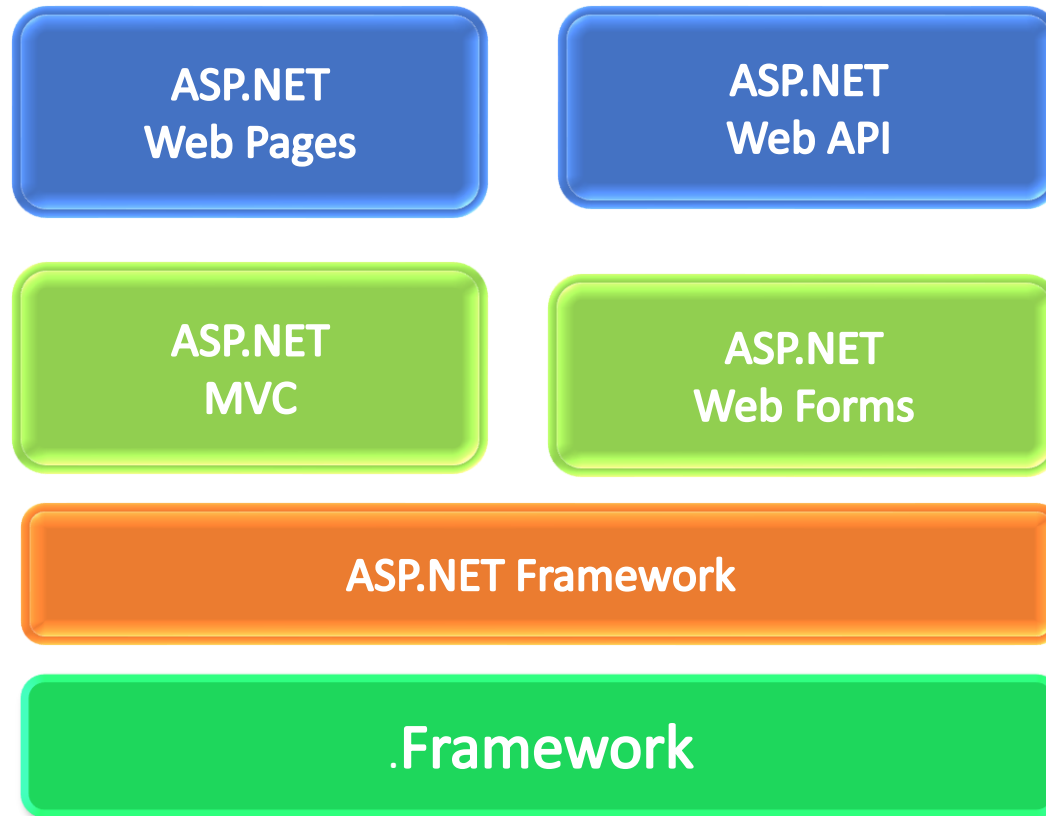
REAL-TIME WEB

SignalR



WHY WE USE DIFFERENT FRAMEWORKS TO BUILD APPLICATIONS?

- 1 Multiple web frameworks built on ASP.NET technology
- 2 All share ASP.NET HTTP run time environment
- 3 Frameworks and can coexist in the same application



Why choose ASP.NET?

Reasons to chose ASP.NET:



- 👉 To create robust website
- 👉 RESTful web Services
- 👉 To incorporate real time functionality in to your apps;

ASP.NET Platform Frameworks

ASP.NET PLATFORM

Which Framework to select ?

1

Type of Application you are building

2

Experience level and skills

3

Development Approach



Web Forms

Web Pages

MVC

SITES

.ASP.NET

ASP.NET Web Forms

- 👉 Build dynamic websites using a library of controls and components
- 👉 That means Drag-and-drop web server controls onto design surface
- 👉 Utilizes event driven programming model

Why Choose ASP.NET Web Forms

- 👉 ASP.NET Rapidly develop UI-driven Websites
- 👉 That means Leverage data controls and simple data access
- 👉 Utilizes event driven programming model

ASP.NET Web Pages

- 👉 ASP.NET Create dynamic web content
- 👉 Combine server code with HTML markup
- 👉 Uses Razor syntax

When to use ASP.NET Web Pages

- 👉 To create small and lightweight sites
- 👉 If you are a beginner to web development
- 👉 If you've worked with a similar development style (ASP, PHP)

ASP.NET MVC

- 👉 Most control and flexibility for building dynamic websites
- 👉 Based on the Model-View-Control design pattern
- 👉 Offers advantages like better testability and mentality
- 👉 Fully extensible framework
- 👉 Advanced development Approach

ASP.NET Web Forms

ASP.NET Web Forms

By Jafar Muzeyin

WEB FORMS

- 👉 Web Forms is a web application framework built on ASP.NET
- 👉 Web Forms provide a suite of web server controls to build your applications.
- 👉 This includes controls that are similar to HTML elements like text inputs and buttons.
- 👉 There are also some more complicated controls that connect to data sources and handle data access.

Name :	<input type="text"/>
Password	<input type="password"/>
Confirm Password	<input type="password"/>
City	<input type="text" value="Select City"/>
Gender	<input type="radio"/> Male <input type="radio"/> Female
Gmail	<input type="text"/>
<input type="button" value="Submit"/>	

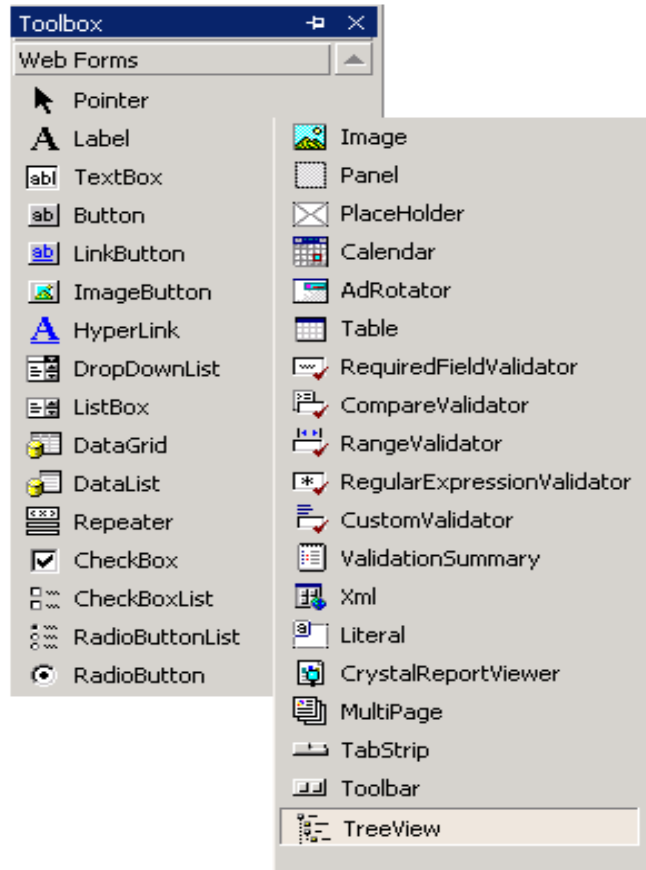
SERVER CONTROLS

- 👉 HTML Server controls are similar to the HTML controls, except they are processed by the server
- 👉 Add `runat = "server"` to the HTML control to transform it into an HTML Server control
- 👉 HTML control: `<input type="text">`
- 👉 HTML Server control:

`<input type="text" runat="server"/>`

`<input type="radio" runat="server" value="Yes"/>`
- 👉 Server-side programs can interact with the control before it is rendered as a plain HTML control and sent to the browser

Controller Server Controls within Visual Studio. Net



👉 In Visual Studio .NET most of the ASP.NET Server controls are located on the Web Forms tab in the toolbox

Server controls with Visual Studio.NET

DIFFERENCES BETWEEN HTML AND ASP.NET CONTROLS

- 👉 HTML loads faster ... better for static pages
- 👉 ASP.NET controls must constantly interact with server
- 👉 HTML control: `<input type="text">`
- 👉 HTML button control is coded in JavaScript:
 - When you double click on it, you are put into HTML source onClick event
 - You then must provide the code for corresponding function e.g. `alert("wellcome")`
- 👉 ASP.NET button control is coded in C# ... when you double click on it creates Event Handler
 - 👉 code behind file ... `aspx.cs`