# EF Interview Questions

top 30 entity framework interview question answer explain

Here are **30 commonly asked Entity Framework interview questions** along with their **answers and explanations**. These questions cover **basic to advanced** concepts:

---

## Basic Level

### 1. What is Entity Framework (EF)?

**Answer:** EF is an open-source ORM (Object-Relational Mapper) developed by Microsoft that enables .NET developers to work with a database using .NET objects, eliminating the need for most data-access code.

### 2. What are the different approaches in EF?

**Answer:**

- **Database-First:** Generates code from an existing database.
- **Model-First:** Creates a model in EF Designer and then generates a database.
- **Code-First:** You write POCO classes, and EF creates the database from them.

### 3. What is a DbContext?

**Answer:** DbContext is a class in EF that manages the connection to the database and provides APIs to perform CRUD operations.

### 4. What is a DbSet?

**Answer:** DbSet represents a collection of entities in the context or database. It is used to query and save instances of entities.

### 5. What is a POCO class?

**Answer:** POCO (Plain Old CLR Object) is a simple class with no inheritance from EF-specific base classes. It's used in Code-First approach.

---

## Intermediate Level

### 6. What is Code First Migrations?

**Answer:** A feature that enables you to incrementally update the database schema as your model changes over time without losing existing data.

### 7. How do you enable Migrations?

**Answer:** Run `` `Enable-Migrations` `` in the Package Manager Console.

### 8. What is Lazy Loading?

**Answer:** Lazy loading is a feature where related data is automatically loaded from the database when it's accessed for the first time.

### 9. What is Eager Loading?

**Answer:** Eager loading is used to load related entities as part of the initial query using `` `.Include()` `` method.

### 10. What is Explicit Loading?

**Answer:** When related data is manually loaded using `` `.Entry(entity).Collection(...).Load()` `` or `` `.Reference(...).Load()` ``.

---

## Performance & Optimization

### 11. What is Change Tracking?

**Answer:** EF automatically tracks changes made to entities and generates appropriate SQL for insert/update/delete operations.

### 12. How can you improve performance in EF?

**Answer:**

- Use `` `.AsNoTracking()` `` for read-only queries
- Use compiled queries
- Minimize use of lazy loading
- Batch queries where possible

### 13. What is `` `.AsNoTracking()` ``?

**Answer:** It disables change tracking for entities returned from a query, which improves performance for read-only operations.

### 14. What is N+1 query problem in EF?

**Answer:** Happens when EF executes a separate query for each related entity, causing performance issues. Solved by eager loading.

### 15. What is a shadow property?

**Answer:** Properties not defined in the .NET class but defined in the EF model. Useful for tracking metadata like timestamps.

---

## Advanced Level

### 16. What is the difference between Add() and Attach()?

**Answer:**

- `` `Add()` `` marks the entity as new and inserts it on `` `SaveChanges()` ``.
- `` `Attach()` `` tells EF to track an existing entity without making changes.

### 17. What is Cascade Delete in EF?

**Answer:** When a parent entity is deleted, all its related child entities are also deleted automatically.

### 18. How do you configure relationships in EF?

**Answer:** Via data annotations (`` `[ForeignKey]` ``, `` `[InverseProperty]` ``) or Fluent API in `` `OnModelCreating()` `` method.

### 19. What is Fluent API?

**Answer:** A way to configure EF model using code instead of attributes, giving more control over model configuration.

### 20. What is the difference between `` `DbContext.SaveChanges()` `` and `` `DbContext.SaveChangesAsync()` ``?

**Answer:** The first is synchronous; the second is asynchronous and doesn't block the main thread.

---

## LINQ & Querying

### 21. What is the difference between IQueryable and IEnumerable in EF?

**Answer:**

- `` `IQueryable` `` builds the query and executes it in the database.
- `` `IEnumerable` `` executes the query in memory after data is loaded.

### 22. How can you execute raw SQL queries in EF?

**Answer:** Use `` `context.Database.SqlQuery<T>()` `` or `` `context.Database.ExecuteSqlCommand()` ``.

### 23. How do you filter and project data using LINQ in EF?

**Answer:** Use LINQ queries or lambda expressions:

```csharp
context.Users.Where(u => u.Age > 25).Select(u => u.Name).ToList();
```

### 24. Can you use stored procedures with EF?

**Answer:** Yes, EF supports stored procedures for CRUD operations and raw queries.

### 25. How to handle concurrency in EF?

**Answer:** Use Concurrency Tokens (like a `` `RowVersion` `` property) to detect and resolve conflicts.

---

## Real-world Use & Miscellaneous

### 26. What is a migration snapshot?

**Answer:** A snapshot is a C# file that stores the current model state. EF uses it to compare changes during migration.

### 27. How do you seed data in EF Core?

**Answer:** In `` `OnModelCreating()` ``, use `` `.HasData()` `` method to provide initial data.

### 28. Can you map a database view in EF?

**Answer:** Yes, use a model class and map it in `` `OnModelCreating()` `` without setting a key.

### 29. What are keyless entities in EF Core?

**Answer:** Entities that do not have a primary key and are used to map to database views or complex queries.

### 30. What is the difference between EF and EF Core?

**Answer:**

- **EF (Entity Framework):** Older, full .NET Framework.
- **EF Core:** Lightweight, cross-platform, modular, and more flexible. It's the modern version.

---

Would you like these in a formatted PDF or with a practice quiz?

top 30 asp.net core interview question asnwer explaination

Here are **30 top ASP.NET Core interview questions** along with their **answers and explanations**, ranging from beginner to advanced level:

---

## ✅ Beginner Level

### 1. What is ASP.NET Core?

**Answer:** ASP.NET Core is a cross-platform, high-performance, open-source web framework developed by Microsoft for building modern web applications, APIs, microservices, and more.

### 2. What are the benefits of using ASP.NET Core?

**Answer:**

- Cross-platform (.NET Core runs on Windows, Linux, macOS)
- Modular architecture
- Built-in dependency injection
- High performance
- Unified MVC and Web API
- Cloud-ready and lightweight

### 3. What is Middleware in ASP.NET Core?

**Answer:** Middleware are software components in the HTTP request pipeline that handle requests and responses (e.g., authentication, routing, CORS).

### 4. What is the Startup class?

**Answer:** The `` `Startup` `` class configures services and the app's request pipeline using `` `ConfigureServices` `` and `` `Configure` `` methods.

### 5. What is the role of `` `Program.cs` `` in ASP.NET Core?

**Answer:** It initializes the app using `` `WebApplication.CreateBuilder()` `` and sets up the hosting environment and app pipeline.

---

## ✅ Middleware & Services

### 6. How do you add middleware to the pipeline?

**Answer:** Middleware is added using `` `app.UseMiddleware<T>()` `` or built-in methods like `` `app.UseRouting()` ``, `` `app.UseAuthentication()` ``.

### 7. What is Dependency Injection (DI)?

**Answer:** A design pattern that allows objects to receive their dependencies via constructor or method injection instead of creating them directly.

### 8. What are the DI lifetimes in ASP.NET Core?

**Answer:**

- `` `Singleton` ``: One instance for the entire app lifetime
- `` `Scoped` ``: One instance per HTTP request
- `` `Transient` ``: New instance each time it's requested

### 9. How do you register services in DI container?

**Answer:** In `` `ConfigureServices` `` method:

```
csharp

services.AddSingleton<IMyService, MyService>();
```

### 10. What is `` `appsettings.json` `` used for?

**Answer:** It stores configuration settings like connection strings, logging, environment settings, and more.

---

## ✅ Routing, Controllers & Views

### 11. What is Routing in ASP.NET Core?

**Answer:** Routing maps incoming HTTP requests to route handlers like controller actions using attribute or conventional routing.

### 12. What is the difference between MVC and Razor Pages?

**Answer:**

- **MVC** uses Controllers and Views.
- **Razor Pages** is a page-focused approach where page and code-behind logic live together.

### 13. What is Tag Helper in Razor views?

**Answer:** Tag helpers allow server-side code to participate in creating and rendering HTML elements, improving readability and maintainability.

### 14. What is ViewModel in MVC?

**Answer:** A ViewModel is a class that contains data you want to display in a view, combining multiple model objects or extra data.

### 15. What is TempData in ASP.NET Core?

**Answer:** TempData is used to pass data between actions and only persists until it's read in the next request (uses session behind the scenes).

---

## ✅ Web API & HTTP Concepts

### 16. How do you create a Web API in ASP.NET Core?

**Answer:** Create a controller that inherits from ``ControllerBase`` and use ``[ApiController]`` and ``[Route("api/[controller]")]`` attributes.

### 17. What is Model Binding?

**Answer:** Model binding maps data from HTTP requests (query string, form, route) to action method parameters.

### 18. What is Model Validation?

**Answer:** Model validation checks data annotations like ``[Required]``, ``[Range]``, and returns validation errors if data is invalid.

### 19. What is CORS?

**Answer:** Cross-Origin Resource Sharing allows or restricts cross-domain AJAX requests. It's configured using ``app.UseCors()``.

### 20. How to handle errors in ASP.NET Core?

**Answer:** Use middleware like ``UseExceptionHandler()`` or ``UseDeveloperExceptionPage()`` and custom error handling in production.

---

## ✅ Authentication & Security

### 21. How does authentication work in ASP.NET Core?

**Answer:** It verifies user identity using schemes like cookies, JWT bearer, or external providers (Google, Facebook).

### 22. How do you implement authorization?

**Answer:** Use ``[Authorize]``, ``[AllowAnonymous]``, or policies to restrict access to actions or controllers.

### 23. What is Identity in ASP.NET Core?

**Answer:** ASP.NET Core Identity is a system for managing users, roles, claims, tokens, and authentication.

### 24. What are claims in ASP.NET Core Identity?

**Answer:** Claims are key-value pairs about a user (like name, role) used for authorization and identity.

### 25. How is password hashing handled in ASP.NET Core Identity?

**Answer:** It uses strong cryptographic algorithms (e.g., PBKDF2) automatically through ``UserManager``.

## ✅ Advanced Topics

### 26. What is Middleware Order and Why is it Important?

**Answer:** Middleware order affects request handling. For example, `` `UseRouting` `` must come before `` `UseAuthorization` ``.

### 27. How to host ASP.NET Core in IIS?

**Answer:** Use the ASP.NET Core Module (ANCM) to forward requests from IIS to Kestrel (built-in web server).

### 28. What is Kestrel?

**Answer:** Kestrel is a cross-platform web server included with ASP.NET Core and used for development and production.

### 29. What is Health Check in ASP.NET Core?

**Answer:** A feature to expose endpoints that monitor app health (`` `/health` ``) for services, DBs, or disk space.

### 30. What is gRPC in ASP.NET Core?

**Answer:** A high-performance RPC framework using HTTP/2 and Protobuf for communication, supported natively in ASP.NET Core.

---

Would you like this content in a PDF, flashcards, or as an interactive quiz to practice?

top 30 asp .net web api interview questions not one liner explain alsos

Here are **30 in-depth ASP.NET Web API interview questions** with **detailed answers and explanations**, ideal for interviews ranging from junior to senior-level roles:

---

## 🔹 1. What is ASP.NET Web API?

**Answer:**
ASP.NET Web API is a framework for building HTTP services that can be consumed by a broad range of clients including browsers, mobile devices, and desktop apps. It is ideal for building RESTful services. Unlike MVC, which returns Views, Web API returns data (typically JSON or XML).

---

## 🔹 2. How is Web API different from MVC?

**Answer:**

| Feature | MVC | Web API |
|---|---|---|
| Return Type | Views (HTML) | Data (JSON/XML) |
| Purpose | Web apps (UI) | Services/REST APIs |
| Routing | Typically uses Views | API-specific routes |
| Format | Formatted for browsers | Consumed by clients/apps |

Web API is designed specifically for data exchange, unlike MVC which is primarily for delivering web pages.

---

### 🔹 3. What is REST?

**Answer:**
REST (Representational State Transfer) is an architectural style for designing web services. It uses standard HTTP methods (GET, POST, PUT, DELETE) and is stateless, meaning each request from the client contains all the information needed for processing.

---

### 🔹 4. What are the HTTP methods used in Web API?

**Answer:**

- **GET** – Retrieve data
- **POST** – Create new data
- **PUT** – Update existing data
- **DELETE** – Delete data
- **PATCH** – Partially update data
  Each method corresponds to an action in a RESTful service.

---

### 🔹 5. What is the difference between PUT and PATCH?

**Answer:**

- **PUT** replaces the entire resource with the new one.
- **PATCH** applies partial modifications to a resource.

Use PUT when updating the whole object, and PATCH for partial updates (e.g., updating just the email).

---

### 🔹 6. How do you create a Web API controller?

**Answer:**
You create a class that inherits from `ApiController` (in older versions) or `ControllerBase` in ASP.NET Core, and decorate it with routing attributes:

```
csharp


[ApiController]
[Route("api/[controller]")]
public class ProductsController : ControllerBase
```

```
{
    [HttpGet]
    public IEnumerable<Product> GetAll() { ... }
}
```

---

### 🔹 7. What is attribute routing?

**Answer:**
Attribute routing allows you to define routes directly above the controller actions using attributes like `` `[Route("api/products/{id}")]` ``.

Example:

```
csharp

[HttpGet("products/{id}")]
public IActionResult GetById(int id) { ... }
```

It gives more control over the URI pattern than conventional routing.

---

### 🔹 8. What is model binding in Web API?

**Answer:**
Model binding maps HTTP request data (from query strings, forms, route values, etc.) to action method parameters. ASP.NET automatically converts JSON/XML into C# objects.

---

### 🔹 9. What is media type formatter in Web API?

**Answer:**
Formatters in Web API determine how the request and response data are serialized. Examples:

- `` `JsonMediaTypeFormatter` ``
- `` `XmlMediaTypeFormatter` ``

You can also create a custom formatter for specific types.

---

### 🔹 10. How does content negotiation work in Web API?

**Answer:**
Content negotiation (conneg) selects the best media type formatter based on the `` `Accept` `` header in the request. For example:

- If the client sends `` `Accept: application/json` ``, Web API returns JSON.
- If `` `application/xml` ``, it returns XML.

---

### 🔹 11. What is dependency injection in Web API?

**Answer:**
Dependency Injection (DI) allows injecting services like repositories or loggers into controllers via

constructors, improving testability and separation of concerns.

```csharp
public ProductsController(IProductService service) { ... }
```

---

### ◆ 12. How do you implement versioning in Web API?

**Answer:**
You can version APIs using:

- **URI versioning:** `` `/api/v1/products` ``
- **Query string:** `` `/api/products?version=1.0` ``
- **Header versioning:** `` `Accept: application/vnd.myapi.v1+json` ``

ASP.NET Core supports API versioning via the `` `Microsoft.AspNetCore.Mvc.Versioning` `` package.

---

### ◆ 13. What are filters in Web API?

**Answer:**
Filters are attributes used to inject logic before/after an action executes. Types include:

- **Authorization filter**
- **Action filter**
- **Exception filter**
- **Result filter**

Example:

```csharp
public class MyLogFilter : ActionFilterAttribute { ... }
```

---

### ◆ 14. What is token-based authentication?

**Answer:**
In token-based auth, a user logs in and receives a token (like JWT), which they send with each request in the `` `Authorization` `` header:

```makefile
Authorization: Bearer <token>
```

It allows stateless, scalable authentication for APIs.

---

### ◆ 15. How do you return HTTP status codes?

**Answer:**

Use helper methods or `` `StatusCode()` ``:

```
csharp

return Ok(data);           // 200
return NotFound();         // 404
return BadRequest();       // 400
return StatusCode(500);    // 500
```

## 🔹 16. What is Swagger in Web API?

**Answer:**

Swagger (OpenAPI) is a tool for generating interactive API documentation. It allows testing endpoints directly from a browser and auto-documents your API structure.

## 🔹 17. What is CORS and how do you enable it?

**Answer:**

CORS (Cross-Origin Resource Sharing) allows requests from different origins. Enable it in `` `Startup.cs` ``:

```
csharp

services.AddCors();
app.UseCors(x => x.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());
```

## 🔹 18. How is exception handling done in Web API?

**Answer:**

You can use:

- `` `try-catch` `` blocks in controllers
- Global exception filters
- `` `UseExceptionHandler()` `` middleware in ASP.NET Core

Example:

```
csharp

app.UseExceptionHandler("/error");
```

## 🔹 19. How do you test Web API?

**Answer:**

- **Postman** or **curl** for manual testing
- **xUnit/NUnit** for unit testing
- Use `` `WebApplicationFactory<T>` `` for integration testing in ASP.NET Core

### 🔹 20. What is IHttpActionResult / IActionResult?

**Answer:**
These are return types that abstract the HTTP response. `` `IActionResult` `` allows you to return various status codes and data in a flexible way:

```
csharp

public IActionResult Get() {
    return Ok(data);
}
```

---

### 🔹 21. What is throttling in Web API?

**Answer:**
Throttling limits the number of requests a user/client can make in a time window, preventing abuse. Implemented via middleware or custom filters.

---

### 🔹 22. What is a DTO?

**Answer:**
A DTO (Data Transfer Object) is a class used to transfer data between layers without exposing domain models directly. Helps maintain encapsulation.

---

### 🔹 23. What is AutoMapper?

**Answer:**
AutoMapper is a library used to map between objects, especially DTOs and domain models.

```
csharp

CreateMap<Product, ProductDto>();
```

---

### 🔹 24. How do you secure a Web API?

**Answer:**

- Use HTTPS
- Use authentication tokens (JWT)
- Validate inputs
- Enable CORS properly
- Rate limiting

---

### 🔹 25. What is HATEOAS in REST?

**Answer:**
HATEOAS (Hypermedia as the Engine of Application State) means the API provides links with responses to guide client actions. Not commonly implemented but part of REST maturity.

---

### ◆ 26. How do you log requests in Web API?

**Answer:**
Use built-in logging (`ILogger<T>`) or middleware to log each incoming request, response, and errors for monitoring and debugging.

---

### ◆ 27. Can Web API return multiple data formats?

**Answer:**
Yes. Content negotiation automatically selects between JSON, XML, etc., based on client `Accept` headers.

---

### ◆ 28. What is async/await and why is it used in Web API?

**Answer:**
`async/await` allows non-blocking I/O operations, making Web API scalable under high load (e.g., async database access).

```
csharp

public async Task<IActionResult> Get() {
    var data = await service.GetAsync();
    return Ok(data);
}
```

---

### ◆ 29. What is a custom action result?

**Answer:**
Custom `IActionResult` allows you to create a reusable response format or logic.

```
csharp

public class CustomResult : IActionResult { ... }
```

---

### ◆ 30. How do you handle file uploads in Web API?

**Answer:**
Use `IFormFile` in ASP.NET Core:

```
csharp

[HttpPost("upload")]
public async Task<IActionResult> Upload(IFormFile file) { ... }
```

Supports uploading files through multipart form-data requests.

Would you like a PDF version or an interactive practice quiz for these questions?

top 30 asp net mvc interview question asnwer not one liner , explain it

Here are the **Top 30 ASP.NET MVC Interview Questions** with **detailed explanations** to help you prepare thoroughly:

---

### 🔹 1. What is ASP.NET MVC?

**Answer:**
ASP.NET MVC is a web application framework developed by Microsoft that implements the **Model-View-Controller (MVC)** architectural pattern. This separates the application into three components:

- **Model**: Handles data and business logic.
- **View**: Displays the UI.
- **Controller**: Handles user interaction and updates the model and view.

It promotes clean separation of concerns, testability, and full control over HTML rendering.

---

### 🔹 2. What are the main advantages of using ASP.NET MVC?

**Answer:**

- Separation of concerns through the MVC pattern.
- Full control over rendered HTML.
- Better support for TDD (Test Driven Development).
- Clean URL routing without file extensions.
- Easily integrates with JavaScript frameworks like Angular or React.

---

### 🔹 3. What is the role of each component in MVC?

**Answer:**

- **Model**: Represents the application's data and business rules.
- **View**: Represents the presentation layer (HTML, CSS, etc.).
- **Controller**: Handles incoming HTTP requests, processes them, and returns appropriate responses.

---

### 🔹 4. What is Routing in ASP.NET MVC?

**Answer:**
Routing maps incoming URLs to controller actions. It allows users to access URLs like

`` `/products/details/5` `` instead of something like `` `/products.aspx?id=5` ``. You configure routes in `` `RouteConfig.cs` ``:

```
csharp

routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
);
```

---

### 🔹 5. What are Action Methods in MVC?

**Answer:**
Action methods are public methods in a controller that respond to HTTP requests. Each action returns a `` `ViewResult` ``, `` `JsonResult` ``, `` `RedirectToRouteResult` ``, etc.

```
csharp

public ActionResult Index() {
    return View();
}
```

---

### 🔹 6. What is Razor in ASP.NET MVC?

**Answer:**
Razor is a markup syntax that allows you to embed C# code into HTML. It’s used to create dynamic web pages in Views:

```
cshtml

<h1>Hello @Model.Name</h1>
```

Razor is clean, compact, and reduces the need for extra syntax like `` `<%= %>` `` in classic ASP.

---

### 🔹 7. What is TempData, ViewData, and ViewBag?

**Answer:**

| Feature | Scope | Type | Purpose |
|---|---|---|---|
| ViewData | Current request | Dictionary | Pass data from controller to view |
| ViewBag | Current request | Dynamic | Easier syntax alternative to ViewData |
| TempData | Across requests | Dictionary | Persist data between redirects |

---

### 🔹 8. What is Partial View?

**Answer:**
A Partial View is a reusable component (like a header, footer, or sidebar) rendered inside another view:

```csharp
@Html.Partial("_Header")
```

It helps reduce redundancy and improves code maintainability.

---

### 🔹 9. What is Layout View?

**Answer:**
Layout View in MVC is similar to a master page in Web Forms. It contains common HTML structure for multiple views, allowing consistency across pages.

```cs
@RenderBody()
```

Used in `` `_Layout.cshtml` ``.

---

### 🔹 10. What is HTML Helper in MVC?

**Answer:**
HTML Helpers are methods that simplify rendering HTML controls in Razor Views:

```csharp
@Html.TextBoxFor(model => model.Name)
@Html.DropDownList("Country", countryList)
```

They improve code readability and reduce manual HTML writing.

---

### 🔹 11. What is Model Binding in MVC?

**Answer:**
Model binding is the process of mapping HTTP request data (form fields, query strings) to parameters or properties in C# objects (models or view models).

```csharp
public ActionResult Save(User user) { ... }
```

MVC handles the conversion automatically.

---

### 🔹 12. What is Data Annotation in MVC?

**Answer:**
Data Annotations are attributes applied to model properties for validation and metadata:

```
csharp

[Required]
[StringLength(50)]
public string Name { get; set; }
```

MVC uses these for model validation on both server and client sides.

---

## ◆ 13. How do you perform server-side validation in MVC?

**Answer:**
Using model state and data annotations:

```
csharp

if (!ModelState.IsValid) {
    return View(model);
}
```

MVC checks the model against validation attributes and sets `` `ModelState.IsValid` ``.

---

## ◆ 14. What is the difference between Server-side and Client-side validation?

**Answer:**

- **Server-side**: Performed on the server, secure, and mandatory.
- **Client-side**: Performed in the browser using JavaScript for faster user feedback.

MVC supports both using unobtrusive JavaScript and validation attributes.

---

## ◆ 15. What is the difference between ViewResult and ActionResult?

**Answer:**

- `` `ActionResult` `` is a base class that supports different return types like `` `ViewResult` ``, `` `JsonResult` ``, `` `RedirectResult` ``.
- `` `ViewResult` `` is a subclass of `` `ActionResult` `` that renders a view.

Returning `` `ActionResult` `` gives flexibility for different return types.

---

## ◆ 16. What is a ViewModel?

**Answer:**
A ViewModel is a custom class that contains data tailored for a specific view. It can combine multiple models or include additional data not in the database.

---

## ◆ 17. What is Scaffolding in MVC?

**Answer:**
Scaffolding automatically generates boilerplate code for CRUD operations based on your models using templates. It saves development time and enforces conventions.

---

### 🔹 18. How is security handled in MVC?

**Answer:**

- **Authentication** (who are you): Forms, Windows, OAuth
- **Authorization** (what can you do): `` `[Authorize]` `` attribute
- **Anti-forgery tokens** to prevent CSRF
- **Input validation** to prevent XSS and SQL Injection

---

### 🔹 19. What is AntiForgeryToken in MVC?

**Answer:**
It prevents CSRF (Cross-Site Request Forgery) attacks by requiring a token in both form and request headers:

```
cshtml

@Html.AntiForgeryToken()
```

Validated with `` `[ValidateAntiForgeryToken]` `` in the controller.

---

### 🔹 20. What are Action Filters in MVC?

**Answer:**
Filters are attributes that allow pre- or post-processing logic around action methods:

- `` `OnActionExecuting` ``
- `` `OnActionExecuted` ``
- `` `OnResultExecuting` ``
- `` `OnResultExecuted` ``

Examples: `` `Authorize` ``, `` `HandleError` ``, `` `OutputCache` ``

---

### 🔹 21. What is Bundling and Minification?

**Answer:**
Bundling combines multiple CSS/JS files into one, and minification removes unnecessary characters. It reduces load time and improves performance.

Use `` `BundleConfig.cs` `` to define bundles.

---

### 🔹 22. How does Dependency Injection work in MVC?

**Answer:**
MVC supports DI via constructor injection. Services or repositories are registered in the container and injected into controllers:

```
csharp

public HomeController(IUserService service) { ... }
```

Improves testability and modularity.

---

## 🔹 23. How to implement custom error handling in MVC?

**Answer:**
Use the `` `[HandleError]` `` attribute, `` `web.config` ``, or global filters. For advanced cases, use custom error pages and override `` `OnException` ``.

---

## 🔹 24. What is the difference between RedirectToAction and Redirect?

**Answer:**

- `` `RedirectToAction` `` redirects to an action method.
- `` `Redirect` `` redirects to a specific URL.

```
csharp

return RedirectToAction("Index");
return Redirect("https://example.com");
```

---

## 🔹 25. What is an Area in MVC?

**Answer:**
Areas help organize large applications into smaller functional sections (e.g., Admin, User). Each area has its own controllers, views, and routes.

---

## 🔹 26. How is AJAX used in MVC?

**Answer:**
AJAX enables partial page updates. Use jQuery to call controller actions and return partial views or JSON:

```
js

$.get("/product/details/5", function(data) {
    $("#result").html(data);
});
```

---

## 🔹 27. What is a strongly typed View?

**Answer:**
A strongly typed view is bound to a specific model using `` `@model` `` at the top. It allows IntelliSense and compile-time checking.

```cshtml
@model MyApp.Models.Product
```

---

## 🔹 28. What is Output Caching?

**Answer:**
Output caching stores the rendered output of a controller action so that future requests are served faster.

```csharp
[OutputCache(Duration = 60)]
public ActionResult Index() { ... }
```

---

## 🔹 29. What are Razor Sections?

**Answer:**
Sections are defined parts of the layout that can be filled in views.

In layout:

```cs
@RenderSection("Scripts", required: false)
```

In view:

```cs
@section Scripts {
    <script src="..."></script>
}
```

---

## 🔹 30. What are the different return types of a Controller Action?

**Answer:**

- `` `ViewResult` `` – renders a view
- `` `JsonResult` `` – returns JSON data
- `` `RedirectResult` `` – redirects to a URL
- `` `RedirectToRouteResult` `` – redirects based on route
- `` `ContentResult` `` – returns plain text
- `` `FileResult` `` – returns a file

Would you like a **PDF**, **Excel format**, or **flashcards** version of this Q&A?

top 50 jquery,ajax, javascript interview question asnwer not one liner , explain it

Here's a comprehensive list of **Top 50 jQuery, AJAX, and JavaScript interview questions** with detailed answers and explanations (not just one-liners):

---

## 🔷 JavaScript – Core Concepts (Q1–Q20)

### 1. What is the difference between `var`, `let`, and `const`?

- `var`: Function-scoped, can be redeclared and updated.
- `let`: Block-scoped, cannot be redeclared in the same scope, but can be updated.
- `const`: Block-scoped, cannot be redeclared or updated. Used for constants.

```js
function test() {
    if (true) {
        var x = 10; // available outside the if block
        let y = 20; // only inside the if block
    }
    console.log(x); // 10
    console.log(y); // ReferenceError
}
```

---

### 2. Explain closures in JavaScript.

A **closure** is a function that has access to its own scope, the outer function's scope, and the global scope, even after the outer function has returned.

```js
function outer() {
    let count = 0;
    return function () {
        count++;
        console.log(count);
    };
}
const counter = outer();
counter(); // 1
counter(); // 2
```

---

### 3. What is hoisting in JavaScript?

Hoisting moves **variable and function declarations** to the top of their scope during compilation. Only declarations are hoisted, not initializations.

```js
console.log(x); // undefined
var x = 5;
```

---

## 4. Difference between `==` and `===`?

- `==`: Compares values after **type coercion**.
- `===`: Compares **value and type** strictly.

```js
'5' == 5 // true
'5' === 5 // false
```

---

## 5. What are arrow functions and how are they different?

Arrow functions:

- Use shorter syntax: `() => {}`.
- Do **not bind their own** `this`.
- Cannot be used as constructors.

```js
const sum = (a, b) => a + b;
```

---

## 6. What is the difference between `null` and `undefined`?

- `undefined`: A variable that has been declared but not assigned.
- `null`: An intentional assignment of "no value".

---

## 7. What is event delegation?

Instead of adding event listeners to many child elements, you add one listener to a parent and use `event.target` to determine the source.

```js
document.getElementById('parent').addEventListener('click', function(e) {
    if (e.target && e.target.matches('button.item')) {
        console.log('Button clicked:', e.target);
    }
});
```

---

## 8. What are promises in JavaScript?

Promises represent **asynchronous operations** that can be in 3 states:

- Pending
- Fulfilled
- Rejected

```js
new Promise((resolve, reject) => {
    resolve("Success");
}).then(console.log).catch(console.error);
```

---

## 9. Explain async/await in JavaScript.

Syntactic sugar over Promises to write cleaner asynchronous code.

```js
async function fetchData() {
    const res = await fetch('url');
    const data = await res.json();
    console.log(data);
}
```

---

## 10. What is the difference between synchronous and asynchronous code?

- **Synchronous**: Executes line-by-line; blocking.
- **Asynchronous**: Executes in the background; non-blocking (e.g., AJAX, `` `setTimeout` ``).

---

## 11. What are higher-order functions?

Functions that take other functions as arguments or return them.

```js
function greet(callback) {
    return callback("Hello");
}
```

---

## 12. Explain prototype and prototypal inheritance.

Every JavaScript object has a prototype from which it can inherit methods and properties. This allows object sharing and inheritance.

```js
function Person(name) {
    this.name = name;
}
Person.prototype.greet = function () {
    return `Hello, ${this.name}`;
};
```

---

## 13. What is the event loop?

The event loop handles **asynchronous callbacks**. It takes callbacks from the task queue and executes them after the call stack is empty.

---

## 14. What is the difference between call, apply, and bind?

- `` `call` ``: Invokes a function with a given `` `this` `` and arguments.
- `` `apply` ``: Same as `` `call` `` but takes arguments as an array.
- `` `bind` ``: Returns a new function with a bound `` `this` ``.

---

## 15. What is destructuring?

Extracting values from arrays or objects:

```
js

const [a, b] = [1, 2];
const { name, age } = { name: "John", age: 30 };
```

---

## 16. What are template literals?

Multi-line strings with embedded expressions:

```
js

let name = "Alice";
console.log(`Hello, ${name}`);
```

---

## 17. What are IIFE (Immediately Invoked Function Expressions)?

Functions that run as soon as defined:

```
js

(function () {
    console.log("IIFE runs");
})();
```

---

## 18. What are pure functions?

Functions that:

- Always return the same output for the same input.

- Have no side effects.

---

## 19. Difference between `` `map` ``, `` `filter` ``, and `` `reduce` ``?

- `` `map` ``: Transforms each element.
- `` `filter` ``: Filters elements.
- `` `reduce` ``: Reduces to a single value.

---

## 20. What is the use of `` `this` `` keyword?

Refers to the context in which a function is called.

---

# 🔷 AJAX Concepts (Q21–Q35)

## 21. What is AJAX?

AJAX (Asynchronous JavaScript and XML) allows **data to be fetched from the server without reloading the page** using `` `XMLHttpRequest` `` or `` `fetch` ``.

---

## 22. What are the main benefits of AJAX?

- Partial page updates
- Improved responsiveness
- Better user experience

---

## 23. How to perform an AJAX call using `` `XMLHttpRequest` ``?

```js
let xhr = new XMLHttpRequest();
xhr.open("GET", "/data", true);
xhr.onload = function() {
    console.log(xhr.responseText);
};
xhr.send();
```

---

## 24. How do you use the Fetch API?

```js
fetch('/api/users')
    .then(res => res.json())
    .then(data => console.log(data))
    .catch(err => console.error(err));
```

---

## 25. Difference between synchronous and asynchronous AJAX requests?

- **Synchronous**: Blocks execution until complete.
- **Asynchronous**: Executes in the background, non-blocking.

---

### 26. What is JSONP?

A technique for **cross-domain AJAX** requests using script tags (only works with GET).

---

### 27. How do you handle errors in AJAX?

Use `` `onerror` `` in `` `XMLHttpRequest` `` or `` `catch` `` in `` `fetch` ``.

---

### 28. How can you cancel an AJAX request?

Using `` `abort()` `` method on `` `XMLHttpRequest` ``.

---

### 29. How to send POST data via AJAX?

```
js

fetch('/submit', {
    method: 'POST',
    headers: {'Content-Type': 'application/json'},
    body: JSON.stringify({name: 'Alice'})
});
```

---

### 30. What is the difference between `` `fetch` `` and `` `XMLHttpRequest` ``?

- `` `fetch` `` is modern, promise-based, and easier to use.
- `` `XMLHttpRequest` `` is older and has more complex syntax.

---

## 🔷 jQuery Concepts (Q36–Q50)

### 31. What is jQuery?

A fast, small, cross-browser JavaScript library that simplifies HTML DOM manipulation, AJAX, and animations.

---

### 32. What are the advantages of jQuery?

- Simplifies JavaScript
- Cross-browser compatibility
- AJAX handling
- Chainable syntax
- Rich plugin ecosystem

---

## 33. How to select elements in jQuery?

```js
$('#id');        // ID
$('.class');     // Class
$('div');        // Element
```

---

## 34. What is chaining in jQuery?

Calling multiple methods on the same element:

```js
$('#box').css('color', 'red').slideUp().slideDown();
```

---

## 35. What is the `ready()` function in jQuery?

Ensures DOM is fully loaded before executing code:

```js
$(document).ready(function() {
    console.log('DOM ready');
});
```

---

## 36. What is the difference between `.html()`, `.text()`, and `.val()`?

- `.html()` gets/sets HTML.
- `.text()` gets/sets plain text.
- `.val()` gets/sets form values.

---

## 37. How to bind events using jQuery?

```js
$('#btn').on('click', function() {
    alert('Clicked');
});
```

---

## 38. How to perform AJAX with jQuery?

```js
$.ajax({
    url: '/api/data',
    method: 'GET',
    success: function(data) {
        console.log(data);
    }
});
```

---

## 39. Difference between `` `.on()` `` and `` `.click()` ``?

- `` `.on()` `` is a general method for binding events (supports delegation).
- `` `.click()` `` is shorthand for binding the click event only.

---

## 40. How to iterate through elements using jQuery?

```js
$('li').each(function(index, element) {
    console.log(index, $(element).text());
});
```

---

## 41. How to add/remove classes in jQuery?

```js
$('#box').addClass('active').removeClass('hidden');
```

---

## 42. What is `` `$.each()` `` in jQuery?

Iterates over arrays or objects:

```js
$.each([1,2,3], function(i, val) {
    console.log(i, val);
});
```

---

## 43. How to fade in/out elements in jQuery?

```js
$('#box').fadeOut();
$('#box').fadeIn();
```

---

## 44. What is the difference between `` `prop()` `` and `` `attr()` ``?

- `` `attr()` ``: Gets/sets HTML attributes.
- `` `prop()` ``: Gets/sets DOM properties (e.g., `` `checked` ``, `` `selected` ``).

---

## 45. How to prevent default action in jQuery?

```js
$('a').click(function(e) {
    e.preventDefault();
});
```

---

## 46. How to delegate events in jQuery?

```js
$('#parent').on('click', '.child', function() {
    console.log('Child clicked');
});
```

---

## 47. What are jQuery effects?

Animations like `.slideUp()`, `.fadeOut()`, `.toggle()` to enhance UI.

---

## 48. How to load data with `.load()`?

```js
$('#result').load('/data.html');
```

---

## 49. What are jQuery selectors?

Powerful expressions to find elements:

- `$('input:checked')`
- `$('div:first')`
- `$('ul li:nth-child(2)')`

---

## 50. How to check if an element exists in jQuery?

```js
if ($('#myDiv').length) {
    console.log('Exists');
}
```

---

Would you like these as a **PDF**, **Excel sheet**, or integrated into a web-based flashcard tool?