Difference between Dispose & Finalize Method

Dispose

Finalize

It is used to free unmanaged resources like files, database connections etc. at any time.

It can be used to free unmanaged resources (when you implement it) like files, database connections etc. held by an object before that object is destroyed.

Explicitly, it is called by user code and the class which is implementing dispose method, must has to implement IDisposable interface.

Internally, it is called by Garbage Collector and cannot be called by user code.

It belongs to IDisposable interface.

It belongs to Object class.

It's implemented by implementing IDisposable interface Dispose() method.

It's implemented with the help of destructor in C++ & C#.

There is no performance costs associated with Dispose method.

There is performance costs associated with Finalize method since it doesn't clean the memory immediately and called by GC automatically.

2. differences between Abstract Class and Interface

Sr. No.	Key	Abstract Class	Interface
1	Definition	In terms of standard definition an Abstract class is, conceptually, a class that cannot be instantiated and is usually implemented as a class that has one or more pure virtual (abstract) functions.	On other hand an Interface is a description of what member functions must a class, which inherits this interface, implement. In other words, an interface describes behaviour of the class.

2	Implementation	As like of other general class design in C# Abstract class also have its own implementation along with its declaration.	On other hand an Interface can only have a signature, not the implementation. While its implementation is being provided by the class which implements it.
3	Inheritance	As per specification in C# a class can extends only one other class hence multiple inheritance is not achieved by abstract class.	On other hand in case of Interface a class can implements multiple interfaces and hence multiple inheritance is achieved by interface.
4	Constructor	Like other classes in C# for instantiation abstract class also have constructor which provide an instance of abstract class to access its non-static methods.	On other hand Interface do not have constructor so we can't instantiate an interface directly although its method could get accessed by creating instance of class which implementing it.
5	Modifiers	As abstract class is most like of other ordinary class in C# so it can contain different types of access modifiers like public, private, protected etc.	On other hand as Interface needs to be get implemented in order to provide its methods implementation by other class so can only contains public access modifier.
6	Performance	As abstract class have its method as well as their implementations also for its abstract methods implementation it	On other hand the performance of interface is slow because it requires time to search actual method in

		have reference for its implementing class so performance is comparatively faster as compare to that of Interface.	the corresponding class.

3 .How do we specify MIME type in Asp.Net WebAPI C#?

A media type, also called a MIME type, identifies the format of a piece of data. In HTTP, media types describe the format of the message body. A media type consists of two strings, a type and a subtype. For example −

- text/html
- image/png
- application/json

When an HTTP message contains an entity-body, the Content-Type header specifies the format of the message body. This tells the receiver how to parse the contents of the message body.

When the client sends a request message, it can include an Accept header. The Accept header tells the server which media type(s) the client wants from the server.

4. **Explain Dependency Injection?**

Answer: Dependency injection is an application design pattern that is implemented by Angular and forms the core concepts of Angular.

Let us understand in a detailed manner. Dependencies in Angular are services which have a functionality. Various components and directives in an application can need these functionalities of the service. Angular provides a smooth mechanism by which these dependencies are injected into components and directives.

5. **difference between the delegate and the event in C#.**

Delegate	Event
A delegate is declared using the delegate keyword.	An event is declared using the event keyword.

Delegate is a function pointer. It holds the reference of one or more methods at runtime.	The event is a notification mechanism that depends on delegates
Delegate is independent and not dependent on events.	An event is dependent on a delegate and cannot be created without delegates. Event is a wrapper around delegate instance to prevent users of the delegate from resetting the delegate and its invocation list and only allows adding or removing targets from the invocation list.
Delegate includes Combine() and Remove() methods to add methods to the invocation list.	EventInfo class inspect events and to hook up event handlers that include methods AddEventHandler() and RemoveEventHandler() methods to add and remove methods to invocation list, respectively.
A delegate can be passed as a method parameter.	An event is raised but cannot be passed as a method parameter.
= operator is used to assigning a single method, and += operator is used to assign multiple methods to a delegate.	= operator cannot be used with events, and only += and -= operator can be used with an event that adds or remove event handler. These methods internally call AddEventHandler and RemoveEventHandler methods.

6. What is the JSON Web Token structure?

In its compact form, JSON Web Tokens consist of three parts separated by dots (.), which are:

- Header
- Payload
- Signature

7. What is CORS?

Ans: CORS also known as Cross-Origin Resource Sharing is a process for accessing various web resources on different domains.

With the help of CORS, web scripts can be integrated more openly with external content of the original domain.

This further leads to better integration between web services.

If you are using Web API 1.0, you can enable CORS support including the following statements in the Application_BeginRequest event handler of the Global.asax.cs file.

```
HttpContext.Current.Response.AddHeader("Access-Control-Allow-Origin",
allowedOrigin);
HttpContext.Current.Response.AddHeader("Access-Control-Allow-Methods",
"G
```

8. Difference between delete, truncate and drop command in SQL

Delete

Delete command is a DML command, it removes rows from a table based on the condition specified in the where clause, being a DML statement we can rollback changes made by delete command.

Truncate

Truncate is a DDL command, it removes all the rows from the table and also frees the space held. It takes a lock on the table while delete command takes a lock on rows of the table.

Drop

Drop is a DDL command, it removes the complete data along with the table structure(unlike truncate command that removes only the rows).

9. Question: How do Observables differ from Promises?

Answer: As soon as a promise is made, the execution takes place. However, this is not the case with observables because they are lazy. This means that nothing happens until a subscription is made. While promises handle

a single event, observable is a stream that allows passing of more than one event. A callback is made for each event in an observable.

10. different ways to communicate between angular components

- @Input – For parent to child communication.
- @ViewChild – For parent to child communication.
- @Output – For child to parent communication using **EventEmitter**.
- service – For any components which are not in parent child structure.

11. Modules – Also known as NgModules, a module is an organized block of code with a specific set of capabilities. It has a specific application domain or a workflow. Like components, any Angular application has at least one module. This is known as the root module. Typically, an Angular application has several modules.

12. Question: What are Lifecycle hooks in Angular? Explain some life cycles hooks.

Answer: Angular components enter its lifecycle from the time it is created to the time it is destroyed. Angular hooks provide ways to tap into these phases and trigger changes at specific phases in a lifecycle.

- **ngOnChanges():** This method is called whenever one or more input properties of the component changes. The hook receives a SimpleChanges object containing the previous and current values of the property.
- **ngOnInit():** This hook gets called once, after the ngOnChanges hook.
- It initializes the component and sets the input properties of the component.
- **ngDoCheck():** It gets called after ngOnChanges and ngOnInit and is used to detect and act on changes that cannot be detected by Angular.
- We can implement our change detection algorithm in this hook.
- **ngAfterContentInit():** It gets called after the first ngDoCheck hook. This hook responds after the content gets projected inside the component.
- **ngAfterContentChecked():** It gets called after ngAfterContentInit and every subsequent ngDoCheck. It responds after the projected content is checked.

- **ngAfterViewInit():** It responds after a component's view, or a child component's view is initialized.
- **ngAfterViewChecked():** It gets called after ngAfterViewInit, and it responds after the component's view, or the child component's view is checked.
- **ngOnDestroy():** It gets called just before Angular destroys the component. This hook can be used to clean up the code and detach event handlers.

13. **What is difference between constructor and ngOnInit?**

The main **difference between constructor and ngOnInit** is that **ngOnInit** is lifecycle hook and runs after **constructor**. Component interpolated template and input initial values aren't available in **constructor**, but they are available in **ngOnInit** . The practical **difference** is how **ngOnInit** affects how the code is structured

14. **Following are some of the important differences between readonly and const keywords.**

Sr. No.	Key	readonly keyword	const keyword
1	Purpose	readonly keyword is used to create a readonly fields.	const keyword is used to create constant fields.
2	Type	readonly is a constant defined at runtime.	const is used to create a constant at compile time.
3	Change	readonly field value can be changed after declaration.	const field value cannot be changed after declaration.
4	Method	readonly fields cannot be defined within a method.	const fields can be declared within a method.
5	Value assignment	readonly variables are declared as instance variable and assigned values in constructor.	const fields are to be assigned at the time of declaration.

15. **Question: Explain Angular Authentication and Authorization.**
Answer: The user login credentials are passed to an authenticate API, which is present on the server. Post server-side validation of the

credentials, a JWT (JSON Web Token) is returned. The JWT has information or attributes regarding the current user. The user is then identified with the given JWT. This is called authentication.

Post logging-in successfully, different users have a different level of access. While some may access everything, access for others might be restricted to only some resources. The level of access is authorization.

Here is a detailed post on Angular 7 - JWT Authentication Example & Tutorial: http://jasonwatmore.com/post/2018/11/16/angular-7-jwt-authentication-example-tutorial

Filter Type	Description	Built-in Filter	Interface
Authorization filters	Performs authentication and authorizes before executing an action method.	[Authorize], [RequireHttps]	IAuthorizationFilter
Action filters	Performs some operation before and after an action method executes.		IActionFilter
Result filters	Performs some operation before or after the execution of the view.	[OutputCache]	IResultFilter
Exception filters	Performs some operation if there is an unhandled exception thrown during the execution of the ASP.NET MVC pipeline.	[HandleError]	IExceptionFilter

- void OnActionExecuted(ActionExecutedContext filterContext)
- void OnActionExecuting(ActionExecutingContext filterContext)