Q1. What is Angular?

Ans: Angular is a TypeScript-based open-source framework developed by Google for building dynamic web applications.

Q2. What is Angular and how is it different from Angular JS?

Ans: Angular is a Typescript-based front-end framework(Angular2+). AngularJS(1.x) is javascript-based and uses controls and scopes, while angular uses components and typescript.

Q3. What is component in Angular?

Ans: A component controls a section of the UI. It uses a typescript class decorated with @Component, a template (HTML), and styles(CSS).

Example:

```
@Component({
  selector: 'app-greet',
  template: `<h1>Hello {{name}}</h1>`
})
export class GreetComponent {
  name = 'IMTIAZ';
}
```

Q4. What is Angular Module?

Ans: Modules(@NgModule) group related components, directives, and services. The root module is AppModule.

Example:

```
@NgModule({
  declarations: [HelloComponent],
  imports: [BrowserModule],
  bootstrap: [HelloComponent]
})
export class AppModule {}
```

Q5. What is directive in Angular?

Ans: Directives are used to manipulate the DOM types:

1. Structural: *ngIf, *ngFor
2. Attribute: ([ngClass], [ngStyle])
3. Custom Directives.

Example:

```
<div *ngIf="isVisible">Visible Content</div>
<ul>
  <li *ngFor="let item of items">{{ item }}</li>
</ul>
```

Q6. What is Angular CLI?

Ans: A command line interface tool to initialize , develop, scaffold and maintain angular apps.

Q7. What is Data Binding in Angular?

Ans: Angular supports one-way and two-way data binding to sync data between model and view.

Q8. What are the different types of data binding?

Ans: Angular is having the following different types of data binding:

1. **Interpolation**: {{value}}
2. **Property Binding**: [property]=”value”
3. **Event Binding:** (event) = “method()”
4. **Two-way Binding:** [(ngModel)]=”value”

Q9. What is ngModel?

Ans: It basically enables two-way binding between the input field and the component property. Requires FormsModule.

Example:

Step1: app.module.ts

```
import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

@NgModule({

  declarations: [AppComponent],

  imports: [BrowserModule, FormsModule],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Step2:

app.component.ts

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  standalone: false, // Required in Angular 19 when using NgModule
  templateUrl: './app.component.html'
})
export class AppComponent {
  name: string = '';
}
```

app.component.html

```
<div style="padding: 20px;">
  <label for="name">Enter your name:</label>
  <input id="name" [(ngModel)]="name" placeholder="Your name" />
  <p>Hello, {{ name }}!</p>
</div>
```

Key Notes:

- FormsModule is imported to enable ngModel.
- standalone: false is explicitly set to allow declaration in NgModule.
- This setup works perfectly in Angular 19 for legacy or modular apps

Q10. What does standalone:true?

Ans:

- It allows the class(component, directive, or pipe) to be used independently.
- It eliminates the need to declare it an @NgModule
- **Example:**

```
import { Component } from '@angular/core';

import { CommonModule } from '@angular/common';

@Component({

selector: 'app-hello',

standalone: true,

imports: [CommonModule],

template: `<p>Hello from a standalone component!</p>`

})

export class HelloComponent {}
```

Q11. Why use standalone components?

Ans:

- It reduces the boilerplate code.

- Improves the performance and modularity.

Q12. What is Directive in Angular?

Ans: A directive is a class that adds behaviour to elements in Angular. It can modify appearance , structure or logic.

Q13. What are the different Directives in Angular?

Ans: The following are the different directives:

1. Component: A directive with a template.

2. Structural: It changes the DOM layout -> *ngIf , *ngFor

3. Attribute: It alternates apperarnace/behaviour : ngClass, ngStyle

Q14. How is a directive different from a component?

Ans: Components have templates and control views. Directives modify existing elements without template.

Q15. How to create our own directive?

Ans: We can use the @Directive() decorator and define the selector.

Example:

```
>ng g directive mydirective
```

mydirective.directive.ts

```
import { Directive, ElementRef, OnInit } from '@angular/core';
@Directive({
  selector: '[myDirective]',
  standalone: true
})
export class MyDirective implements OnInit {
  constructor(private el: ElementRef) {}
  ngOnInit() {
    this.el.nativeElement.style.backgroundColor = 'green';
  }
}
```

app.component.ts

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
 standalone: true,
  imports:[MyDirective]
  templateUrl: './app.component.html'
})
export class AppComponent {
  title = 'Angular 19 Custom Directive';
}
app.component.html
<div myDirective style="padding: 20px; color: white;">
  This div has a green background thanks to myDirective!
</div>
```

Q16. What is the role of @Input() and @Output?

Ans:

@Input() : It allows parent to pass data to child

@Output(): It allows the child to emits events to parent via EventEmiiter

Example:

app.component.s

```
import { Component } from '@angular/core';
import { ChildComponent } from './child.component';
@Component({
  selector: 'app-root',
  standalone: true,
  imports: [ChildComponent],
  template: `
    <app-child [message]="parentMessage" (reply)="handleReply($event)"></app-child>
    <p>Reply from child: {{ childReply }}</p>
  `
})
export class AppComponent {
  parentMessage = 'Hi Child!';
  childReply = '';
  handleReply(event: string) {
    this.childReply = event;
  }
}
```

>ng g childcomponent

child.component.ts

```
import { Component, Input, Output, EventEmitter } from '@angular/core';
@Component({
 selector: 'app-child',
 standalone: true,
 template: `
  <p>Message from parent: {{ message }}</p>
  <button (click)="sendReply()">Reply to Parent</button>
 `
})
export class ChildComponent {
 @Input() message: string = '';
 @Output() reply = new EventEmitter<string>();

 sendReply() {
  this.reply.emit('Hello Parent!');
 }
}
```

Q17. What is ngClass in Angular?

Ans: In Angular, ngClass is a build- in directive used to dynamically assign or remove one or more CSS classes to an element based on an expression or condition.

In enhance the static class attribute by supporting conditional, array-based or object-based binding.

Example:

app.component.ts

```
import {Component} from '@angular/core';
@Component({
selector: 'app-toggle-button',
templateUrl:'./toggle-button.component.html',
styleUrls:['./toggle-button.component.css']
})
export class ToggleButtonComponent{
isActive:Boolean=false;
toggleStatus(){
this.isActive =!this.isActive;
}
}
```

toggle-button.component.html

```
<button (click)="toggleStatus()" [ngClass]="{'btn-succes':isActive, 'btn-danger':!isActive}">
 {{isActive?'Active':'InActive'}}
</button>
```

toggle-button.component.css

```
.btn-success{
background-color:green;
color:white;
padding:5px;
border:none;
}
.btn-danger{
background-color:red;
color:white;
padding:5px;
border:none;
}
```

Q18. What is ngStyle in Angular?

Ans: Is a build-in Angular directive used to apply inline CSS styles dynamically to an HTML element.

It works like the HTML style attribute but supports Angular expressions , so the styles can change based on conditions.

Example:

status-indicator.component.ts

```
import {Component}  from '@angular/core';

@Component({

selector: 'app-status-indicator',

templateUrl: './status-indicator.component.html',

})

export class StatusIndicatorComponent{

isOnline:Boolean=true;

toggleStatus(){

this.isOnline=!this.isOnline;

}

}
```

status-indicator.component.html

```
<p [ngStyle]="{

              'color': isOnline? 'green': 'red',

              'font-size':20px

}">

{{isOnline? 'User is Online' : 'User is offline'}}

</p>

<button (click)="toggleStatus">Toggle Status</button>
```

Example2:

Inside the app.component.ts

```
boxStyle={
'border' : '1px solid red',
'background-color':'black',
'color': 'white',
'margin.px':10,
};
```

Inside the app.component.html

```
<div [ngStyle]="boxStyle">
<p>Hello , World!</p>
</div>
```

Q19. Difference between ngStyle and ngClass?

ngStyle	ngClass
Applies inline styles	Applies CSS classes
Accepts object literal or expression	It accepts strings, array, or object.
It can use where style elements based on dynamic conditions	It can be used where add/remove classes dynamically.
It renders in style=""	It renders in class=""

Q20. Can we bind styles dynamically without using ngStyle?

Ans : Yes by using [style.color]="value" or [style.fontSize.px]="size"

Q22. Can we use ngStyle and ngClass together?

Ans : Yes, ngStyle for inline styles, and ngClass for class-based styling.

Q24. What is template-driven forms?

Ans :