Question 1

What is the correct syntax for importing the `HttpClientModule` in an Angular module?

```typescript
import { HttpClientModule } from
'@angular/common/http';
```

A)

```typescript
import { HttpClientModule } from '@angular/http';
```

B)

```typescript
import { HttpClient } from
'@angular/common/http';
```

C)

```typescript
import { HttpClientModule } from '@angular/common/http';
```

D)

```typescript
import { HttpModule } from '@angular/common/http';
```

Answer: C

Question 2

Which decorator is used to define a service in Angular?

A)

```typescript
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
```

B)

```typescript
@NgModule({
  declarations: [],
  imports: [],
  providers: []
})
```

C)

```typescript
@Directive({
  selector: '[appDirective]'
})
```

D)

```typescript
@Injectable({
  providedIn: 'root'
})
```

Answer: D

Question 3

How do you bind a property in Angular using two-way data binding?

A)

```html
<input [(ngModel)]="property">
```

B)

```html
<input [ngModel]="property">
```

C)

```html
<input {{property}}>
```

D)

```html
<input (ngModel)="property">
```

Answer: A

Question 4

What is the purpose of the `ngOnInit` lifecycle hook in Angular?

A) To initialize a component's properties.

B) To define a component's metadata.

C) To execute logic when a component is destroyed.

D) To handle input properties changes.

Answer: A

Question 5

Which of the following is correct for creating an Angular component using the Angular CLI?

A)

```bash
ng create component my-component
```

B)

```bash
ng generate component my-component
```

C)

```bash
ng new component my-component
```

D)

```bash
ng make component my-component
```

Answer: B

Question 6

How do you define a route in Angular?

A)

```typescript
const routes: Routes = [
  { path: '', component: HomeComponent }
];
```

B)

```typescript
const routes: Routes = [
  { path: '', template: HomeComponent }
];
```

C)

```typescript
const routes: Routes = [
  { url: '', component: HomeComponent }
];
```

D)

```typescript
const routes: Routes = [
  { path: '', templateUrl: HomeComponent }
];
```

Answer: A

Question 7

What is the correct way to inject a service into an Angular component?

A)

```typescript
constructor(private service: MyService) {}
```

B)

```typescript
constructor(service: MyService) {}
```

C)

```typescript
constructor(inject service: MyService) {}
```

D)

```typescript
constructor(public service: MyService) {}
```

Answer: A

Question 8

Which of the following commands is used to start an Angular development server?

A)

```bash
ng start
```

B)

```bash
ng serve
```

C)

```bash

ng run

```

D)

```bash

ng dev

```

Answer: B

Question 9

What is the purpose of Angular's `HttpClient` service?

A) To handle component lifecycle events.

B) To facilitate HTTP requests.

C) To create Angular modules.

D) To define application routes.

Answer: B

Question 10

Which Angular directive is used to conditionally include an element in the DOM?

A)

```html
<div *ngIf="condition">...</div>
```

B)

```html
<div *ngFor="let item of items">...</div>
```

C)

```html
<div *ngSwitch="expression">...</div>
```

D)
```html
<div *ngModel="model">...</div>
```

Answer: A
Question 11
How do you create a reactive form in Angular?

A)
```typescript
this.form = this.formBuilder.group({
  name: ['']
});
```

B)

```typescript
this.form = this.formBuilder.form({
  name: ['']
});
```

C)

```typescript
this.form = this.formBuilder.control({
  name: ['']
});
```

D)

```typescript
this.form = this.formBuilder.array({
  name: ['']
});
```

Answer: A

Question 12

Which pipe is used to transform a string to uppercase in Angular?

A)

```html
{{ value | lower }}
```

B)

```html
{{ value | titlecase }}
```

C)

```html
{{ value | uppercase }}
```

D)

```html
{{ value | capitalize }}
```

Answer: C

Question 13

How do you register a service provider in an Angular module?

A)

```typescript
@NgModule({
  declarations: [],
  imports: [],
  providers: [MyService]
})
```

B)

```typescript
```

~~~
@NgModule({
  declarations: [MyService],
  imports: [],
  providers: []
})
```
~~~

C)

~~~
```typescript
@NgModule({
  declarations: [],
  imports: [MyService],
  providers: []
})
```
~~~

D)

~~~
```typescript
@NgModule({
  declarations: [],
  imports: [],
~~~

  services: [MyService]
})

Answer: A

Question 14
What is the correct syntax for using an Angular directive in a template?

A)
```html
<div ngDirective></div>
```

B)
```html
<div *ngDirective></div>
```

C)

```html
<div [ngDirective]="expression"></div>
```

D)
```html
<div (ngDirective)="expression"></div>
```

Answer: B

Question 15
How do you create an event binding in Angular?

A)
```html
<button (click)="doSomething()"></button>
```

B)
```html
```

<button [click]="doSomething()"></button>

C)
```html
<button {click}="doSomething()"></button>
```

D)
```html
<button *click="doSomething()"></button>
```

Answer: A

Question 16
Which lifecycle hook is called when Angular has finished initializing all data-bound properties of a directive?

A) `ngOnChanges`

B) `ngOnInit`

C) `ngDoCheck`

D) `ngAfterViewInit`

Answer: B

Question 17
How do you define a custom pipe in Angular?

A)
```typescript
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({name: 'myPipe'})
export class MyPipe implements PipeTransform {
  transform(value: string): string {
    return value.toUpperCase();
  }
}
```

B)

```typescript
import { Directive, DirectiveTransform } from '@angular/core';

@Directive({name: 'myPipe'})
export class MyPipe implements DirectiveTransform {
  transform(value: string): string {
    return value.toUpperCase();
  }
}
```

C)

```typescript
```

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

@Component({name: 'myPipe'})
export class MyPipe implements PipeTransform {
  transform(value: string): string {
    return value.toUpperCase();
  }
}
```

D)

```typescript
import { Module, PipeTransform } from
'@angular/core';

@Module({name: 'myPipe'})
export class MyPipe implements PipeTransform {
  transform(value: string): string {
    return value.toUpperCase();
  }
```

}

Answer: A

Question 18

Which command is used to create a new Angular service using the Angular CLI?

A)

```bash
ng generate service my-service
```

B)

```bash
ng create service my-service
```

C)

```bash
```

ng make service my-service

D)
```bash
ng add service my-service
```

Answer: A

Question 19
How do you access route parameters in Angular?

A)
```typescript
this.route.params.subscribe(params => {
  console.log(params['id']);
});
```

B)

```typescript
this.route.queryParams.subscribe(params => {
  console.log(params['id']);
});
```

C)

```typescript
this.route.fragment.subscribe(params => {
  console.log(params['id']);
});
```

D)

```typescript
this.route.data.subscribe(params => {
  console.log(params['id']);
});
```

Answer: A

Question 20

What is the purpose of the `RouterModule.forRoot` method?

A) To configure the root module of an Angular application.

B) To set up the application’s root routes.

C) To initialize Angular services.

D) To declare Angular components.

Answer: B

Question 21

Which directive is used to repeat a template for each item in a list?

A)

```html
<div *ngIf="item of items"></div>
```

B)

```html
<div *ngFor="let item of items"></div>
```

C)

```html
<div *ngRepeat="item of items"></div>
```

D)

```html
<div *ngEach="item of items"></div>
```

Answer: B

Question 22

How do you handle HTTP errors in Angular using HttpClient?

A)

```typescript
this.http.get('url').catchError(error => {
  console.error(error);
});
```

B)

```typescript
this.http.get('url').subscribe(data => {}, error => {
  console.error(error);
});
```

C)

```typescript
this.http.get('url').handleError(error => {
```

  console.error(error);
});

D)
```typescript
this.http.get('url').onError(error => {
  console.error(error);
});
```

Answer: B

Question 23
What is the correct way to import the `FormsModule` in an Angular module?

A)
```typescript
import { FormsModule } from '@angular/forms';
```

B)

```typescript
import { FormsModule } from '@angular/common/forms';
```

C)

```typescript
import { FormsModule } from '@angular/http';
```

D)

```typescript
import { FormsModule } from '@angular/platform-browser/forms';
```

Answer: A

Question 24

How do you make an Angular service singleton?

A)
```typescript
@Injectable({
  providedIn: 'root'
})
export class MyService {}
```

B)
```typescript
@Injectable({
  providedIn: 'app'
})
export class MyService {}
```

C)
```typescript
@Injectable({
```

  providedIn: 'module'
})
export class MyService {}

D)
```typescript
@Injectable({
  providedIn: 'singleton'
})
export class MyService {}
```

Answer: A

Question 25

Which Angular directive is used to bind styles to an element?

A)

```html
```

<div [ngStyles]="styles"></div>

B)
```html
<div [style]="styles"></div>
```

C)
```html
<div [ngClass]="styles"></div>
```

D)
```html
<div [ngStyle]="styles"></div>
```

Answer: D

Question 26

How do you pass data from a parent component to a child component in Angular?

A)
```typescript
@Input() childProperty: string;
```

B)
```typescript
@Output() childProperty: string;
```

C)
```typescript
@Child() childProperty: string;
```

D)
```typescript
@Parent() childProperty: string;
```

Answer: A

Question 27

Which command is used to build an Angular application for production?

A)

```bash
ng build --prod
```

B)

```bash
ng serve --prod
```

C)

```bash
ng deploy --prod
```

D)

```bash
ng make --prod
```

Answer: A

Question 28

How do you subscribe to an observable in Angular?

A)

```typescript
observable.subscribe(value => {
  console.log(value);
});
```

B)

```typescript
observable.watch(value => {
  console.log(value);
});
```

C)

```typescript
observable.listen(value => {
  console.log(value);
});
```

D)

```typescript
observable.handle(value => {
  console.log(value);
});
```

Answer: A

Question 29

What is the purpose of Angular's `RouterLink` directive?

A) To navigate to a different route programmatically.

B) To handle HTTP requests.

C) To define a link to a different route.

D) To bind data to a template.

Answer: C

Question 30

Which command is used to add a new Angular library to an existing project?

A)

```bash
```

```
ng add @angular/library
```

B)

```bash
ng generate library my-library
```

C)

```bash
ng new library my-library
```

D)

```bash
ng create library my-library
```

Answer: B

Question 31

How do you create a custom validator in Angular?

A)
```typescript
function myValidator(control: FormControl) {
  return control.value ? null : { required: true };
}
```

B)
```typescript
function myValidator(control: FormControl) {
  return control.value ? { required: true } : null;
}
```

C)
```typescript
function myValidator(control: FormControl) {
  return control.value ? { required: true } : {
```

 valid: true };
}

D)
```typescript
function myValidator(control: FormControl) {
  return control.value ? { valid: true } : { required: true };
}
```

Answer: A

Question 32
Which lifecycle hook is called after the component's view has been fully initialized?

A) `ngAfterContentInit`

B) `ngAfterContentChecked`

C) `ngAfterViewInit`

D) `ngAfterViewChecked`

Answer: C

Question 33
How do you add a CSS class to an element conditionally in Angular?

A)
```html
<div [ngClass]="{ 'active': isActive }"></div>
```

B)
```html
<div [class]="{ 'active': isActive }"></div>
```

C)

```html
<div [ngStyle]="{ 'active': isActive }"></div>
```

D)

```html
<div [style]="{ 'active': isActive }"></div>
```

Answer: A

Question 34

How do you create a custom Angular directive?

A)

```typescript
import { Directive } from '@angular/core';

@Directive({
  selector: '[appCustom]'
})
export class CustomDirective {}
```

B)

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

@Component({
  selector: '[appCustom]'
})
export class CustomDirective {}
```

C)

```typescript
import { Injectable } from '@angular/core';

@Injectable({
  selector: '[appCustom]'
})
export class CustomDirective {}
```

D)

```typescript
import { Pipe } from '@angular/core';

@Pipe({
  selector: '[appCustom]'
})
export class CustomDirective {}
```

Answer: A

Question 35

Which Angular directive is used to iterate over a list of items?

A)

```html
<div *ngIf="item of items"></div>
```

B)

```html
<div *ngFor="let item of items"></div>
```

C)

```html
<div *ngRepeat="item of items"></div>
```

D)

```html
<div *ngEach="item of items"></div>
```

Answer: B

Question 36

How do you register a route with parameters in Angular?

A)

```typescript
const routes: Routes = [
  { path: 'details/:id', component: DetailsComponent }
];
```

B)

```typescript
const routes: Routes = [
  { path: 'details/{id}', component: DetailsComponent }
];
```

C)

```typescript
const routes: Routes = [
  { path: 'details?{id}', component: DetailsComponent }
];
```

D)

```typescript
const routes: Routes = [
  { path: 'details#id', component: DetailsComponent }
];
```

Answer: A

Question 37

How do you perform content projection in Angular?

A)

```html
<ng-template>
  <ng-content></ng-content>
</ng-template>
```

B)

```html
<ng-container>
  <ng-content></ng-content>
</ng-container>
```

C)

```html
<ng-content></ng-content>
```

D)

```html
<ng-project></ng-project>
```

Answer: C

Question 38

Which Angular CLI command is used to create a new module?

A)

```bash

ng new module my-module

```

B)

```bash

ng generate module my-module

```

C)

```bash

ng create module my-module
```

D)

```bash
ng add module my-module
```

Answer: B

Question 39

How do you create a nested route in Angular?

A)

```typescript
const routes: Routes = [
  { path: 'parent', component: ParentComponent, children: [
    { path: 'child', component: ChildComponent }
  ]}
];
```

B)

```typescript
const routes: Routes = [
  { path: 'parent/child', component: ChildComponent }
];
```

C)

```typescript
const routes: Routes = [
  { path: 'parent', component: ParentComponent, children: [
    { path: '', component: ChildComponent }
  ]}
];
```

D)

```typescript
```

```
const routes: Routes = [
  { path: 'parent', component: ParentComponent, children: [
    { path: ':child', component: ChildComponent }
  ]}
];
```

Answer: A

Question 40

What is the purpose of Angular's `Renderer2` service?

A) To manipulate the DOM elements.

B) To handle HTTP requests.

C) To manage Angular forms.

D) To define Angular routes.

Answer: A

Question 41

How do you perform form validation in Angular?

A)

```html
<form [formGroup]="form" (ngSubmit)="onSubmit()">
  <input formControlName="name" required>
</form>
```

B)

```html
<form [formGroup]="form" (submit)="onSubmit()">
  <input formControlName="name" required>
</form>
```

C)

```html
<form [form]="form" (ngSubmit)="onSubmit()">
  <input formControlName="name" required>
</form>
```

D)

```html
<form [formControl]="form" (ngSubmit)="onSubmit()">
  <input formControlName="name" required>
</form>
```

Answer: A

Question 42

How do you configure lazy loading in Angular?

A)

```typescript
const routes: Routes = [
  { path: 'lazy', loadChildren: () => import('./lazy/lazy.module').then(m => m.LazyModule) }
];
```

B)

```typescript
const routes: Routes = [
  { path: 'lazy', loadChildren: () => import('./lazy.module').then(m => m.LazyModule) }
];
```

C)

```typescript
const routes: Routes = [
  { path: 'lazy', loadChildren: './lazy/lazy.module#LazyModule' }
];
```

D)

```typescript
const routes: Routes = [
  { path: 'lazy', loadChildren: './lazy.module#LazyModule' }
];
```

Answer: A

Question 43

Which lifecycle hook is called when a component's content has been fully initialized?

A) `ngAfterContentInit`

B) `ngAfterViewInit`

C) `ngOnChanges`

D) `ngDoCheck`

Answer: A

Question 44
How do you pass multiple values to an Angular pipe?

A)
```html
{{ value | myPipe: param1, param2 }}
```

B)
```html
{{ value | myPipe: param1:param2 }}
```

C)
```html
```

```
{{ value | myPipe: [param1, param2] }}
```

D)

```html
{{ value | myPipe(param1, param2) }}
```

Answer: B

Question 45

What is the purpose of Angular's `ChangeDetectionStrategy`?

A) To optimize the performance of Angular applications by controlling change detection.

B) To handle HTTP requests.

C) To manage Angular forms.

D) To define Angular routes.

Answer: A

Question 46
How do you use Angular's `ViewChild` decorator?

A)
```typescript
@ViewChild('myElement') myElement: ElementRef;
```

B)
```typescript
@ViewChild({ static: true }) myElement: ElementRef;
```

C)
```typescript
```

@ViewChild(ElementRef) myElement: ElementRef;

D)

```typescript
@ViewChild('myElement', { static: true })
myElement: ElementRef;
```

Answer: D

Question 47

How do you configure global styles in an Angular application?

A) In the `angular.json` file, under the `styles` array.

B) In the `index.html` file, within a `<style>` tag.

C) In the `app.module.ts` file, under the `styles` array.

D) In the `main.ts` file, within a `<style>` tag.

Answer: A

Question 48

Which Angular decorator is used to define a module?

A)

```typescript
@Component({
  declarations: [],
  imports: [],
  providers: []
})
```

B)

```typescript
@Directive({
  selector: '[appDirective]'
```

})

C)
```typescript
@Pipe({
  name: 'appPipe'
})
```

D)
```typescript
@NgModule({
  declarations: [],
  imports: [],
  providers: []
})
```

Answer: D

Question 49

What is the purpose of Angular's `HttpInterceptor`?

A) To handle component lifecycle events.

B) To intercept and modify HTTP requests and responses.

C) To create Angular modules.

D) To define application routes.

Answer: B

Question 50

How do you define a custom event in Angular?

A)

```typescript
```

@Output() customEvent = new EventEmitter<string>();

B)

```typescript
@Input() customEvent = new EventEmitter<string>();
```

C)

```typescript
@ViewChild() customEvent = new EventEmitter<string>();
```

D)

```typescript
@Directive() customEvent = new EventEmitter<string>();
```

Answer: A

Question 51

How do you apply conditional CSS classes to an element in Angular?

A)

```html
<div [ngClass]="{ 'class1': condition1, 'class2': condition

2 }"></div>
```

B)

```html
<div [class]="{ 'class1': condition1, 'class2': condition2 }"></div>
```

C)

```html
<div [ngStyle]="{ 'class1': condition1, 'class2': condition2 }"></div>
```

D)
```html
<div [style]="{ 'class1': condition1, 'class2': condition2 }"></div>
```

Answer: A

Question 52
How do you create a shared module in Angular?

A)
```typescript
@NgModule({
  declarations: [SharedComponent],
  imports: [CommonModule],
```

  exports: [SharedComponent]
})
export class SharedModule {}

B)

```typescript
@NgModule({
  declarations: [],
  imports: [CommonModule],
  providers: [SharedComponent]
})
export class SharedModule {}
```

C)

```typescript
@NgModule({
  declarations: [SharedComponent],
  imports: [CommonModule],
  providers: [SharedComponent]
```

})
export class SharedModule {}

D)
```typescript
@NgModule({
  declarations: [],
  imports: [CommonModule],
  exports: [SharedComponent]
})
export class SharedModule {}
```

Answer: A

Question 53
What is the purpose of Angular's `FormBuilder` service?

A) To handle HTTP requests.

B) To create reactive forms.

C) To manage Angular routes.

D) To define Angular modules.

Answer: B

Question 54

How do you perform dependency injection in Angular?

A)

```typescript
constructor(private service: MyService) {}
```

B)

```typescript
constructor(service: MyService) {}
```

C)

```typescript
constructor(inject service: MyService) {}
```

D)

```typescript
constructor(public service: MyService) {}
```

Answer: A

Question 55

How do you create a unit test for an Angular service?

A)

```typescript
import { TestBed } from '@angular/core/testing';
import { MyService } from './my-service.service';

describe('MyService', () => {
  let service: MyService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.inject(MyService);
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
});
```

B)

```typescript
import { TestBed } from '@angular/core/testing';
import { MyService } from './my-service.service';
```

```typescript
describe('MyService', () => {
  let service: MyService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.injector(MyService);
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
});
```

C)

```typescript
import { TestBed } from '@angular/core/testing';
import { MyService } from './my-service.service';

describe('MyService', () => {
  let service: MyService;
  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.injector(MyService);
  });

  it('should be created', () => {
    expect(service).toBeTrue();
  });
});
```

D)

```typescript
import { TestBed } from '@angular/core/testing';
import { MyService } from './my-service.service';

describe('MyService', () => {
  let service: MyService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.inject(MyService);
  });

  it('should be created', () => {
    expect(service).toBeTrue();
  });
});
```

Answer: A

Question 56

How do you listen for changes to an input property in Angular?

A)

```typescript
ngOnChanges(changes: SimpleChanges) {
  if (changes['inputProperty']) {
```

console.log(changes['inputProperty'].currentValue);
  }
}

B)
```typescript
ngOnInit(changes: SimpleChanges) {
  if (changes['inputProperty']) {

console.log(changes['inputProperty'].currentValue);
  }
}
```

C)
```typescript
ngDoCheck(changes: SimpleChanges) {
  if (changes['inputProperty']) {
```

```
console.log(changes['inputProperty'].currentValue);
  }
}
```

D)

```typescript
ngAfterViewInit(changes: SimpleChanges) {
  if (changes['inputProperty']) {

console.log(changes['inputProperty'].currentValue);
  }
}
```

Answer: A

Question 57

How do you create a feature module in Angular?

A)

```typescript
@NgModule({
  declarations: [FeatureComponent],
  imports: [CommonModule],
  exports: [FeatureComponent]
})
export class FeatureModule {}
```

B)

```typescript
@NgModule({
  declarations: [],
  imports: [CommonModule],
  providers: [FeatureComponent]
})
export class FeatureModule {}
```

C)

```typescript
@NgModule({
  declarations: [FeatureComponent],
  imports: [CommonModule],
  providers: [FeatureComponent]
})
export class FeatureModule {}
```

D)

```typescript
@NgModule({
  declarations: [],
  imports: [CommonModule],
  exports: [FeatureComponent]
})
export class FeatureModule {}
```

Answer: A

Question 58

How do you handle route guards in Angular?

A)

```typescript
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree } from '@angular/router';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    return true;
  }
}
```

B)

```typescript
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree } from '@angular/router';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    return false;
  }
}
```

C)

```typescript
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree } from '@angular/router';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    return false;
  }
}
```

D)

```typescript
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree } from '@angular/router';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    return true;
  }
```

}

Answer: A

Question 59
How do you set default values for form controls in Angular?

A)
```typescript
this.form = this.formBuilder.group({
  name: ['default value']
});
```

B)
```typescript
this.form = this.formBuilder.group({
  name: ['']
});
```

C)
```typescript
this.form = this.formBuilder.group({
  name: [null]
});
```

D)
```typescript
this.form = this.formBuilder.group({
  name: [undefined]
});
```

Answer: A

Question 60
What is the purpose of Angular's `RouterLinkActive` directive?

A) To apply a CSS class to an element when its corresponding route is active.

B) To handle HTTP requests.

C) To manage Angular forms.

D) To define Angular routes.

Answer: A

Question 61

How do you create a dynamic component in Angular?

A)

```typescript
import { Component, ComponentFactoryResolver, ViewChild, ViewContainerRef } from '@angular/core';
```

```
@Component({
  selector: 'app-dynamic',
  template: '<ng-template #container></ng-template>'
})
export class DynamicComponent {
  @ViewChild('container', { read: ViewContainerRef, static: true }) container: ViewContainerRef;

  constructor(private resolver: ComponentFactoryResolver) {}

  loadComponent() {
    const factory = this.resolver.resolveComponentFactory(ChildComponent);
    this.container.createComponent(factory);
  }
}
```

B)

```typescript
import { Component, ComponentFactoryResolver, ViewChild, ViewContainerRef } from '@angular/core';

@Component({
  selector: 'app-dynamic',
  template: '<ng-template #container></ng-template>'
})
export class DynamicComponent {
  @ViewChild('container', { read: ViewContainerRef, static: true }) container: ViewContainerRef;

  constructor(private resolver: ComponentFactoryResolver) {}

  loadComponent() {
    const factory = this.resolver.resolveComponentFactory(ChildComponent);
    this.container.insert(factory);
  }
}
```

C)

```typescript
import { Component, ComponentFactoryResolver, ViewChild, ViewContainerRef } from '@angular/core';

@Component({
  selector: 'app-dynamic',
  template

: '<ng-template #container></ng-template>'
})
export class DynamicComponent {
  @ViewChild('container', { read: ViewContainerRef, static: true }) container: ViewContainerRef;

  constructor(private resolver: ComponentFactoryResolver) {}
  loadComponent() {
    const factory = this.resolver.resolveComponentFactory(ChildComponent);
    this.container.add(factory);
  }
}
```

D)

```typescript
import { Component, ComponentFactoryResolver, ViewChild, ViewContainerRef } from '@angular/core;

@Component({
  selector: 'app-dynamic',
  template: '<ng-template #container></ng-template>'
})
export class DynamicComponent {
  @ViewChild('container', { read: ViewContainerRef, static: true }) container: ViewContainerRef;

  constructor(private resolver: ComponentFactoryResolver) {}

  loadComponent() {
    const factory = this.resolver.resolveComponentFactory(ChildComponent);
    this.container.append(factory);
  }
}
```

Answer: A

Question 62

Which Angular decorator is used to define an injectable service?

A)

```typescript
@Injectable({
  providedIn: 'root'
})
export class MyService {}
```

B)

```typescript
@Directive({
  providedIn: 'root'
})
export class MyService {}
```

C)

```typescript
@Component({
  providedIn: 'root'
})
export class MyService {}
```

D)

```typescript
@Pipe({
  providedIn: 'root'
})
export class MyService {}
```

Answer: A

Question 63

How do you configure a custom theme in Angular Material?

A) By defining a custom theme in the `styles.scss` file.

B) By modifying the `angular.json` file.

C) By creating a custom theme module.

D) By configuring the `app.module.ts` file.

Answer: A

Question 64

How do you create a reusable component in Angular?

A)

```typescript
@Component({
  selector: 'app-reusable',
  templateUrl: './reusable.component.html',
  styleUrls: ['./reusable.component.css']
})
export class ReusableComponent {}
```

B)

```typescript
@Directive({
  selector: 'app-reusable',
  templateUrl: './reusable.component.html',
  styleUrls: ['./reusable.component.css']
})
export class ReusableComponent {}
```

C)

```typescript
@Injectable({
  selector: 'app-reusable',
  templateUrl: './reusable.component.html',
  styleUrls: ['./reusable.component.css']
})
export class ReusableComponent {}
```

D)

```typescript
```

@Pipe({
  selector: 'app-reusable',
  templateUrl: './reusable.component.html',
  styleUrls: ['./reusable.component.css']
})
export class ReusableComponent {}

Answer: A

Question 65

How do you use Angular's `HttpClient` to send a POST request?

A)

```typescript
this.http.post('url', data).subscribe(response => {
  console.log(response);
});
```

B)

```typescript
this.http.post('url', data).then(response => {
  console.log(response);
});
```

C)

```typescript
this.http.post('url', data).get(response => {
  console.log(response);
});
```

D)

```typescript
this.http.post('url', data).fetch(response => {
  console.log(response);
});
```

Answer: A

Question 66

How do you handle asynchronous operations in Angular?

A)

```typescript
this.http.get('url').subscribe(response => {
  console.log(response);
});
```

B)

```typescript
this.http.get('url').then(response => {
  console.log(response);
});
```

C)

```typescript
this.http.get('url').await(response => {
  console.log(response);
});
```

D)
```typescript
this.http.get('url').fetch(response => {
  console.log(response);
});
```

Answer: A

Question 67
How do you create a route guard in Angular?

A)
```typescript
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree } from '@angular/router';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    return true;
  }
}
```

B)

```typescript
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree } from '@angular/router';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    return false;
  }
}
```

C)

```typescript
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree } from '@angular/router';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    return false;
  }
}
```

D)

```typescript
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree } from '@angular/router';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    return true;
  }
}
```

Answer: A

Question 68

How do you set up Angular Material in an Angular project?

A) By running `ng add @angular/material` in the terminal.

B) By running `ng generate @angular/material` in the terminal.

C) By running `ng install @angular/material` in the terminal.

D) By running `ng create @angular/material` in the terminal.

Answer: A

Question 69

How do you create a custom Angular pipe?

A)

```typescript
```

~~~
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'customPipe'
})
export class CustomPipe implements PipeTransform {
  transform(value: string): string {
    return value.toUpperCase();
  }
}
```

B)
```typescript
import { Directive, PipeTransform } from '@angular/core;

@Directive({
  name: 'customPipe'
~~~

~~~
})
export class CustomPipe implements PipeTransform {
  transform(value: string): string {
    return value.toUpperCase();
  }
}
```
~~~

C)

~~~
```typescript
import { Component, PipeTransform } from '@angular/core;

@Component({
  name: 'customPipe'
})
export class CustomPipe implements PipeTransform {
  transform(value: string): string {
    return value.toUpperCase();
~~~

```
  }
}
```

D)

```typescript
import { Injectable, PipeTransform } from '@angular/core;

@Injectable({
  name: 'customPipe'
})
export class CustomPipe implements PipeTransform {
  transform(value: string): string {
    return value.toUpperCase();
  }
}
```

Answer: A

Question 70

How do you perform HTTP GET requests in Angular?

A)

```typescript
this.http.get('url').subscribe(response => {
  console.log(response);
});
```

B)

```typescript
this.http.fetch('url').subscribe(response => {
  console.log(response);
});
```

C)

```typescript
```

this.http.post('url').subscribe(response => {
  console.log(response);
});

D)
```typescript
this.http.put('url').subscribe(response => {
  console.log(response);
});
```

Answer: A

Question 71
How do you use Angular's `Router` service to navigate to a different route?

A)
```typescript
this.router.navigate(['/path']);
```

B)
```typescript
this.router.go(['/path']);
```

C)
```typescript
this.router.route(['/path']);
```

D)
```typescript
this.router.move(['/path']);
```

Answer: A

Question 72
How do you create a child route in Angular?

A)

```typescript
const routes: Routes = [
  { path: 'parent', component: ParentComponent, children: [
    { path: 'child', component: ChildComponent }
  ]}
];
```

B)

```typescript
const routes: Routes = [
  { path: 'parent/child', component: ChildComponent }
];
```

C)

```typescript
```

const routes: Routes = [
  { path: 'parent', component: ParentComponent, children: [
    { path: '', component: ChildComponent }
  ]}
];

D)

```typescript
const routes: Routes = [
  { path: 'parent', component: ParentComponent, children: [
    { path: ':child', component: ChildComponent }
  ]}
];
```

Answer: A

Question 73

How do you use Angular's `ActivatedRoute` to access query parameters?

A)

```typescript
this.route.queryParams.subscribe(params => {
  console.log(params['id']);
});
```

B)

```typescript
this.route.params.subscribe(params => {
  console.log(params['id']);
});
```

C)

```typescript
this
```

.route.data.subscribe(params => {
  console.log(params['id']);
});

D)
```typescript
this.route.fragment.subscribe(params => {
  console.log(params['id']);
});
```

Answer: A

Question 74
How do you define a custom event in Angular?

A)
```typescript
@Output() customEvent = new
EventEmitter<string>();
```

B)

```typescript
@Input() customEvent = new EventEmitter<string>();
```

C)

```typescript
@ViewChild() customEvent = new EventEmitter<string>();
```

D)

```typescript
@Directive() customEvent = new EventEmitter<string>();
```

Answer: A

Question 75

How do you create a new Angular component using the Angular CLI?

A)

```bash
ng generate component my-component
```

B)

```bash
ng create component my-component
```

C)

```bash
ng add component my-component
```

D)

```bash
ng make component my-component
```

Answer: A

Question 76

How do you import Angular's `CommonModule` in a feature module?

A)

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

B)

```typescript
import { CommonModule } from '@angular/core';
```

C)

```typescript
import { CommonModule } from '@angular/forms';
```

D)

```typescript
import { CommonModule } from '@angular/router';
```

Answer: A

Question 77

How do you configure a route with a resolver in Angular?

A)

```typescript
const routes: Routes = [
  { path: 'details', component: DetailsComponent, resolve: { data: DataResolver } }
];
```

B)

```typescript
const routes: Routes = [
  { path: 'details', component: DetailsComponent, resolve: { data: DataResolverService } }
];
```

C)

```typescript
const routes: Routes = [
  { path: 'details', component: DetailsComponent, resolve: { data: Resolver } }
];
```

D)

```typescript
```

const routes: Routes = [
  { path: 'details', component: DetailsComponent, resolve: { data: DetailsResolver } }
];

Answer: A

Question 78

How do you use Angular's `Renderer2` service to set an attribute on an element?

A)

```typescript
this.renderer.setAttribute(element, 'attr', 'value');
```

B)

```typescript
this.renderer.setProperty(element, 'attr', 'value');
```

C)

```typescript
this.renderer.setStyle(element, 'attr', 'value');
```

D)

```typescript
this.renderer.setElementAttribute(element, 'attr', 'value');
```

Answer: A

Question 79

How do you perform HTTP DELETE requests in Angular?

A)

```typescript
this.http.delete('url').subscribe(response => {
  console.log(response);
});
```

B)

```typescript
this.http.remove('url').subscribe(response => {
  console.log(response);
});
```

C)

```typescript
this.http.post('url').subscribe(response => {
  console.log(response);
});
```

D)

```typescript
this.http.put('url').subscribe(response => {
```

  console.log(response);
});

Answer: A

Question 80

How do you use Angular's `Renderer2` service to add a class to an element?

A)

```typescript
this.renderer.addClass(element, 'class-name');
```

B)

```typescript
this.renderer.setClass(element, 'class-name');
```

C)

```typescript
this.renderer.pushClass(element, 'class-name');
```

D)
```typescript
this.renderer.insertClass(element, 'class-name');
```

Answer: A

Question 81
How do you set default values for form controls in Angular?

A)
```typescript
this.form = this.formBuilder.group({
  name: ['default value']
});
```

B)

```typescript
this.form = this.formBuilder.group({
  name: ['']
});
```

C)

```typescript
this.form = this.formBuilder.group({
  name: [null]
});
```

D)

```typescript
this.form = this.formBuilder.group({
  name: [undefined]
});
```

}

Answer: A

Question 82

How do you use Angular's `HostListener` decorator to handle DOM events?

A)

```typescript
@HostListener('eventName', ['$event'])
handleEvent(event: Event) {
  console.log(event);
}
```

B)

```typescript
@HostListener('eventName', ['event'])
handleEvent(event: Event) {
  console.log(event);
}
```

C)
```typescript
@HostListener('eventName', ['$event'])
handleEvent(event) {
  console.log(event);
}
```

D)
```typescript
@HostListener('eventName', ['event'])
handleEvent(event) {
  console.log(event);
}
```

Answer: A

Question 83

How do you configure a custom theme in Angular Material?

A) By defining a custom theme in the `styles.scss` file.

B) By modifying the `angular.json` file.

C) By creating a custom theme module.

D) By configuring the `app.module.ts` file.

Answer: A

Question 84

How do you use Angular's `ViewChildren` decorator to query multiple elements?

A)

```typescript
@ViewChildren('myElement') myElements: QueryList<ElementRef>;
```

B)

```typescript
@ViewChildren('myElement') myElements: ElementRef[];
```

C)

```typescript
@ViewChildren('myElement') myElements: Array<ElementRef>;
```

D)

```typescript
@ViewChildren('myElement') myElements: Array<QueryList<ElementRef>>;
```

Answer: A

Question 85

How do you use Angular's `HostBinding` decorator to bind properties to the host element?

A)

```typescript
@HostBinding('class.active') isActive = true;
```

B)

```typescript
@HostBinding('attr.active') isActive = true;
```

C)

```typescript
@HostBinding('style.active') isActive = true;
```

D)

```typescript
@HostBinding('class.active') isActive = false;
```

Answer: A

Question 86

How do you configure global styles in an Angular application?

A) In the `angular.json` file, under the `styles` array.

B) In the `index.html` file, within a `<style>` tag.

C) In the `app.module.ts` file, under the `styles` array.

D) In the `main.ts` file, within a `<style>` tag.

Answer: A

Question 87

How do you create a custom Angular directive?

A)

```typescript
import { Directive } from '@angular/core';

@Directive({
  selector: '[appCustom]'
})
export class CustomDirective {}
```

B)

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

@Component({
  selector: '[appCustom]'
})
export class CustomDirective {}
```

C)

```typescript
import { Injectable } from '@angular/core;

@Injectable({
  selector: '[appCustom]'
})
export class CustomDirective {}
```

D)

```typescript
import { Pipe } from '@angular/core;

@Pipe({
  selector: '[appCustom]'
})
export class CustomDirective {}
```

Answer: A

Question 88

How do you use Angular's `ElementRef` service to access a DOM element?

A)

```typescript
constructor(private el: ElementRef) {}
```

B)

```typescript
constructor(public el: ElementRef) {}
```

C)

```typescript
constructor(protected el: ElementRef) {}
```

D)

```typescript
constructor(readonly el: ElementRef) {}
```

Answer: A

Question 89

How do you handle HTTP errors in Angular using HttpClient?

A)

```typescript
this.http.get('url').subscribe(data => {}, error => {
  console.error(error);
});
```

B)

```typescript
this.http.get('url').catchError(error => {
  console.error(error);
});
```

C)

```typescript
this.http.get('url').handleError(error => {
  console.error(error);
});
```

D)

```typescript
this.http.get('url').onError(error => {
  console.error(error);
});
```

Answer: A

Question 90

How do you use Angular's `Router` service to navigate to a different route programmatically?

A)
```typescript
this.router.navigate(['/path']);
```

B)
```typescript
this.router.go(['/path']);
```

C)
```typescript
this.router.route(['/path']);
```

D)
```typescript
this.router.move(['/path']);
```

Answer: A

Question 91
How do you create a nested route in Angular?

A)
```typescript
const routes: Routes = [
  { path: 'parent', component: ParentComponent, children: [
    { path: 'child', component: ChildComponent }
  ]}
];
```

B)
```typescript
const routes: Routes = [
  { path: 'parent/child', component: ChildComponent }
];
```

C)

```typescript
const routes: Routes = [
  { path: 'parent', component: ParentComponent, children: [
    { path: '', component: ChildComponent }
  ]}
];
```

D)

```typescript
const routes: Routes = [
  { path: 'parent', component: ParentComponent, children: [
    { path: ':child', component: ChildComponent }
```

  ]}
];
``

`

Answer: A

Question 92
How do you use Angular's `ActivatedRoute` to access route parameters?

A)
```typescript
this.route.params.subscribe(params => {
  console.log(params['id']);
});
```

B)
```typescript
```

```
this.route.queryParams.subscribe(params => {
  console.log(params['id']);
});
```

C)

```typescript
this.route.data.subscribe(params => {
  console.log(params['id']);
});
```

D)

```typescript
this.route.fragment.subscribe(params => {
  console.log(params['id']);
});
```

Answer: A

Question 93

How do you use Angular's `HttpClient` to send a GET request?

A)

```typescript
this.http.get('url').subscribe(response => {
  console.log(response);
});
```

B)

```typescript
this.http.fetch('url').subscribe(response => {
  console.log(response);
});
```

C)

```typescript
this.http.post('url').subscribe(response => {
```

  console.log(response);
});

D)
```typescript
this.http.put('url').subscribe(response => {
  console.log(response);
});
```

Answer: A

Question 94
How do you create a custom Angular pipe?

A)
```typescript
import { Pipe, PipeTransform } from
'@angular/core;
```

```
@Pipe({
  name: 'customPipe'
})
export class CustomPipe implements PipeTransform
{
  transform(value: string): string {
    return value.toUpperCase();
  }
}
```

B)

```typescript
import { Directive, PipeTransform } from
'@angular/core;

@Directive({
  name: 'customPipe'
})
export class CustomPipe implements PipeTransform
{
  transform(value: string): string {
    return value.toUpperCase();
  }
}
```

C)

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

@Component({
  name: 'customPipe'
})
export class CustomPipe implements PipeTransform {
  transform(value: string): string {
    return value.toUpperCase();
  }
}
```

D)

```typescript
import { Injectable, PipeTransform } from '@angular/core;

@Injectable({
  name: 'customPipe'
})
export class CustomPipe implements PipeTransform {
  transform(value: string): string {
    return value.toUpperCase();
  }
}
```

Answer: A

Question 95

How do you use Angular's `HostListener` decorator to handle DOM events?

A)

```typescript
@HostListener('eventName', ['$event'])
handleEvent(event: Event) {
  console.log(event);
}
```

B)

```typescript
@HostListener('eventName', ['event'])
handleEvent(event: Event) {
  console.log(event);
}
```

C)

```typescript
```

```
@HostListener('eventName', ['$event'])
handleEvent(event) {
  console.log(event);
}
```

D)

```typescript
@HostListener('eventName', ['event'])
handleEvent(event) {
  console.log(event);
}
```

Answer: A

Question 96

How do you configure a route with a resolver in Angular?

A)

```typescript
const routes: Routes = [
  { path: 'details', component: DetailsComponent, resolve: { data: DataResolver } }
];
```

B)

```typescript
const routes: Routes = [
  { path: 'details', component: DetailsComponent, resolve: { data: DataResolverService } }
];
```

C)

```typescript
const routes: Routes = [
  { path: 'details', component: DetailsComponent, resolve: { data: Resolver } }
];
```

D)

```typescript
const routes: Routes = [
  { path: 'details', component: DetailsComponent, resolve: { data: DetailsResolver } }
];
```

Answer: A

Question 97

How do you use Angular's `Renderer2` service to set an attribute on an element?

A)

```typescript
this.renderer.setAttribute(element, 'attr', 'value');
```

B)

```typescript
this.renderer.setProperty(element, 'attr', 'value');
```

C)

```typescript
this.renderer.setStyle(element, 'attr', 'value');
```

D)

```typescript
this.renderer.setElementAttribute(element, 'attr', 'value');
```

Answer: A

Question 98

How do you perform HTTP DELETE requests in Angular?

A)

```typescript
this.http.delete('url').subscribe(response => {
  console.log(response);
});
```

B)

```typescript
this.http.remove('url').subscribe(response => {
  console.log(response);
});
```

C)

```typescript
this.http.post('url').subscribe(response => {
  console.log(response);
});
```

D)

```typescript
this.http.put('url').subscribe(response => {
  console.log(response);
});
```

Answer: A

Question 99

How do you create a dynamic component in Angular?

A)

```typescript
import { Component, ComponentFactoryResolver, ViewChild, ViewContainerRef } from '@angular/core';

@Component({
  selector: 'app-dynamic',
  template: '<ng-template #container></ng-template>'
})
export class DynamicComponent {
  @ViewChild('container', { read: ViewContainerRef, static: true }) container: ViewContainerRef;

  constructor(private resolver: ComponentFactoryResolver) {}

  loadComponent() {
    const factory = this.resolver.resolveComponentFactory(ChildComponent);
    this.container.createComponent(factory);
  }
}
```

B)

```typescript
```

```
import { Component, ComponentFactoryResolver, ViewChild, ViewContainerRef } from '@angular/core';

@Component({
  selector: 'app-dynamic',
  template: '<ng-template #container></ng-template>'
})
export class DynamicComponent {
  @ViewChild('container', { read: ViewContainerRef, static: true }) container: ViewContainerRef;

  constructor(private resolver: ComponentFactoryResolver) {}

  loadComponent() {
    const factory = this.resolver.resolveComponentFactory(ChildComponent);
    this.container.insert(factory);
  }
}
```

C)

```typescript
import { Component, ComponentFactoryResolver, ViewChild, ViewContainerRef } from '@angular/core';

@Component({
  selector: 'app-dynamic',
  template: '<ng-template #container></ng-template>'
})
export class DynamicComponent {
  @ViewChild('container', { read: ViewContainerRef, static: true }) container: ViewContainerRef;

  constructor(private resolver: ComponentFactoryResolver) {}

  loadComponent() {
    const factory = this.resolver.resolveComponentFactory(ChildComponent);
    this.container.add(factory);
  }
}
```

D)

```typescript
import { Component, ComponentFactoryResolver, ViewChild, ViewContainerRef } from '@angular/core;

@Component({
  selector: 'app-dynamic',
  template: '<ng-template #container></ng-template>'
})
export class DynamicComponent {
  @ViewChild('container', { read: ViewContainerRef, static: true }) container: ViewContainerRef;
```

  constructor(private resolver: ComponentFactoryResolver) {}

  loadComponent() {
    const factory = this.resolver.resolveComponentFactory(ChildComponent);
    this.container.append(factory);
  }
}
~~~

Answer: A

Question 100
How do you use Angular's `Router` service to navigate to a different route programmatically?

A)
```typescript
this.router.navigate(['/path']);
~~~

B)

```typescript
this.router.go(['/path']);
```

C)

```typescript
this.router.route(['/path']);
```

D)

```typescript
this.router.move(['/path']);
```

Answer: A