1. Configuration de Projet

- **Créer un projet Angular :**

```
ng new nom-du-projet
cd nom-du-projet
ng serve  # Pour lancer l'application localement
```

- **Structure de base :**
  - `src/app/` : Contient les composants, services, modules.
  - `src/assets/` : Stocke les ressources statiques.
  - `angular.json` : Fichier de configuration pour Angular CLI.

2. Modules

- **AppModule :** module principal du projet (`app.module.ts`).

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

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

3. Composants

- **Création d'un composant :**

```
ng generate component nom-du-composant
```

- **Structure de base d'un composant :**

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

@Component({
  selector: 'app-nom-du-composant',
  templateUrl: './nom-du-composant.component.html',
  styleUrls: ['./nom-du-composant.component.css']
})
export class NomDuComposant {
  public titre: string = 'Bonjour Angular!';
}
```

4. Binding (liaison)

- **Interpolation :** `{{ data }}` pour afficher des données.
- **Property binding :** `[property]="expression"` .
- **Event binding :** `(event)="method()"` .
- **Two-way binding :** `[()]` avec `[(ngModel)]` (nécessite `FormsModule`).

5. Directives

- **Directives structurelles :**
  - `*ngIf` , `*ngFor` , `*ngSwitchCase`
- **Directives d'attribut :**
  - `[ngClass]` , `[ngStyle]`

6. Services et Injection de Dépendances

- **Créer un service :**

```
ng generate service nom-du-service
```

- **Définition et injection d'un service :**

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

@Injectable({
  providedIn: 'root'
})
export class NomDuService {
  getData() { /* ... */ }
}
```

7. Observables

- **Déclaration d'un Observable** :

```
import { Observable, of } from 'rxjs';

const monObservable = new Observable((observer) => {
  observer.next('Hello');
  observer.complete();
});
```

- **Utilisation d'un Observable** :

```
monObservable.subscribe({
  next: (value) => console.log(value),
  complete: () => console.log('Complete')
});
```

- **Observables HTTP avec `HttpClient`** :

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

constructor(private http: HttpClient) { }

getData() {
  return this.http.get('https://api.example.com/data');
}
```

8. Router

- **Configuration des routes** :

```
import { RouterModule, Routes } from '@angular/router';

const routes: Routes = [
  { path: 'home', component: HomeComponent },
  { path: 'about', component: AboutComponent },
  { path: '', redirectTo: '/home', pathMatch: 'full' }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

- **Navigation dans le template** :

```
<a routerLink="/about">About</a>
```

9. Formulaires

- **Formulaires basés sur les modèles (Template-Driven Forms)** :

```
import { FormsModule } from '@angular/forms';
@NgModule({ imports: [FormsModule] })
export class AppModule { }
```

```
<form #f="ngForm" (ngSubmit)="onSubmit(f.value)">
  <input name="username" ngModel required />
  <button type="submit">Submit</button>
</form>
```

- **Formulaires réactifs** :

```
import { ReactiveFormsModule, FormGroup, FormControl } from '@angular/forms';

this.form = new FormGroup({
  name: new FormControl(''),
  email: new FormControl('')
});
```

10. Pipes

- **Utilisation de pipes** :

```
{{ dateVariable | date:'short' }}
```

- **Création d'un pipe personnalisé** :

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

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

11. Intercepteurs HTTP

- **Création d'un intercepteur** :

```
import { Injectable } from '@angular/core';
import { HttpInterceptor, HttpRequest, HttpHandler } from '@angular/common/http';

@Injectable()
export class AuthInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler) {
    const cloned = req.clone({
      headers: req.headers.set('Authorization', 'Bearer token')
    });
    return next.handle(cloned);
  }
}
```

12. Tests

- **Tests unitaires avec Jasmine** :

```
describe('NomDuComposant', () => {
  it('devrait être créé', () => {
    const component = new NomDuComposant();
    expect(component).toBeTruthy();
  });
});
```

- **Tests d'intégration avec Angular TestBed** :

```
import { TestBed } from '@angular/core/testing';
TestBed.configureTestingModule({ declarations: [NomDuComposant] }).compileComponents();
```