Angular Core Concepts (with Examples)

This guide serves as a comprehensive introduction to some of the most essential Angular concepts used in building a dynamic employee list module. It includes real-world examples to illustrate how Angular facilitates powerful front-end development using data binding, components, and structural directives.

1. Data Binding in Angular

Data binding is the mechanism that allows communication between the component class and the template (HTML). Angular provides various ways to bind data:

Two-Way Binding – `[(ngModel)]`

- Synchronizes data between the input element and component variable.
- You type in the input box, the variable updates, and vice versa.
- Needs `FormsModule` to be imported.

Example:

```
<input type="text" [(ngModel)]="searchTerm" placeholder="Search by
name" />
<p>{{ searchTerm }}</p> <!-- Binds variable to template -->
```

Property Binding – `[property]`

- Binds the value of a property in the component to a property of a DOM element.
- Useful for setting element attributes dynamically.

Example:

```
<img [src]="employee.imageUrl" [alt]="employee.name" />
<td [style.color]="employee.yearsOfExperience > 5 ? 'green' : 'gray'">
  {{ employee.role }}
```

```
</td>
```

Event Binding – (event)

- Executes a method when an event occurs.
- Mostly used for click, input, submit events.

Example:

```
<button (click)="clearSearch()">Clear Search</button>
clearSearch() {
  this.searchTerm = '';
}
```

2. Angular Components

Angular components are the core building blocks of an application. Each component has:

- A class for logic (TypeScript)
- An HTML template
- Optional CSS for styles

AppComponent

- Root component where the application starts.

```
<app-employee></app-employee>
```

EmployeeComponent

- Acts as a parent component and container for employee-related modules.

```
<app-employee-lists></app-employee-lists>
```

EmployeeListComponent

- Responsible for displaying, searching, and filtering employee data.

```
employeesData = [
  { name: 'Alice', role: 'Associate Engineer', yearsOfExperience: 2 },
  { name: 'Bob', role: 'Senior Engineer', yearsOfExperience: 3 },
  { name: 'Nicole', role: 'Principal Engineer', yearsOfExperience: 7 }
];
```

3. Directives

Directives add behavior to elements or components. There are structural and attribute directives.

*ngFor – Loop through a list

- Dynamically creates DOM elements based on an array.

```
<tr *ngFor="let employee of filteredEmployees()">
  <td>{{ employee.name }}</td>
</tr>
```

*ngIf – Conditional rendering

- Displays or hides elements based on a condition.

```
<p *ngIf="filteredEmployees().length === 0">
  No Employee Data Found
</p>
```

Ternary Operator

- Inline conditional check to show status based on experience.

```
<td>{{ employee.yearsOfExperience === 0 ? 'Fresher' :
'Experienced' }}</td>
```

4. Filtering Logic

This method filters employees based on the input search term. It is triggered automatically due to two-way binding.

```
filteredEmployees() {
  return this.employeesData.filter(emp =>
    emp.name.toLowerCase().includes(this.searchTerm.toLowerCase())
  );
}
```

Important: Always ensure the function `filter` has a return statement.

5. Helpful Tips for Module Setup

- Always import `FormsModule` in the module to use `ngModel`.

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

- Declare and export components correctly:

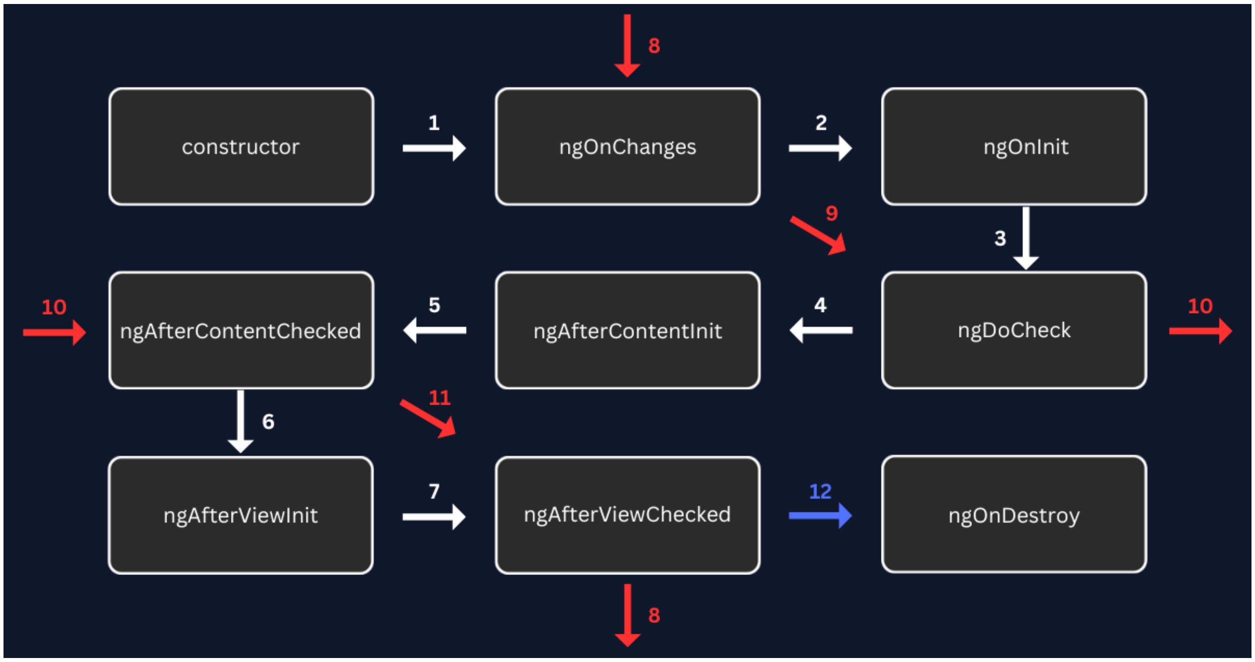
```
@NgModule({
  declarations: [EmployeeComponent, EmployeeListComponent],
  imports: [CommonModule, FormsModule],
  exports: [EmployeeComponent],
})
export class EmployeeModule {}
```

Summary of Concepts Covered

Concept	Description & Example
Two-Way Binding	`[(ngModel)]` with search input
Property Binding	`[src]`, `[style.color]`
Event Binding	`(click)` event on button
Component Usage	Modular component structure
*ngFor Directive	Table rows for each employee
*ngIf Directive	Display message when data is empty
Ternary Operator	Conditional text: Fresher vs Experienced

Extra:

Angular Lifecyle :


8
constructor
1
ngOnChanges
2
ngOnInit
9
3
10
ngAfterContentChecked
5
ngAfterContentInit
4
ngDoCheck
10
11
6
ngAfterViewInit
7
ngAfterViewChecked
12
ngOnDestroy
8