Chandrakanth Avula

What is Dependency Injection in Angular ?

Dependancy Injection

Chandrakanth Avula
@chandrakanth1

Scenario with Example

Let’s imagine a **School with classroom:**

Every day, the teacher needs things like chalk, markers, and books to teach the students. Now, what if the teacher had to bring all these things from home? It would be tiring and take a lot of time!

So, the school helps out. They have a Utilities Room with all the supplies. The teacher just asks for what they need, and the school gives it to them. Now, the teacher can focus on teaching, not worrying about bringing supplies.

This is how **Dependency Injection (DI)** works in Angular: Instead of every part of your app (like teachers) finding or managing their own tools, Angular gives them everything they need. This makes the app work smoothly, just like the classroom runs better when the school provides the supplies!

Why Do We Need Dependency Injection?

Imagine if every Teacher in the school:

- Had to bring their own chalk, books, and whiteboards.
- It would waste time and resources!

The same happens in apps:

- If every component had to create its own tools (services), your app would **get messy and hard to manage.**

Above Example

School is the App : because The overall system managing components and their tools (services).

Teachers are Components : Perform specific tasks like teaching, similar to components handling UI or logic.

Chalk, Books & Whiteboards are Services : Tools provided to components for reusable functionality or data.

how does Angular solve this problem? Let's find out!

What is Dependency Injection?

In Angular, **Dependency Injection works like the school principal:**

- The principal provides the resources (services) to the teachers (components).
- Teachers don’t worry about where resources come from — they just use them!

But where does Angular store these school supplies ?

Let’s take a look 👉👉👉

Providers – The School Supply Closet

Imagine **a supply closet in a school with chalk, books, and markers.** Teachers don't bring their own supplies, **they get them from the closet.**

In Angular, this **supply closet** is called a **Provider**. It gives your app's components what they need.

For example: 👉👉👉

```
1  @Injectable({ providedIn: 'root' })
2  export class ChalkService {
3    // Chalk-related logic
4  }
```

Here, **@Injectable({ providedIn: 'root' })** means the service is available to the whole app, just like a school supply closet is shared by all classrooms. Components can "ask" for this service when needed, making your app organized and efficient!

Injecting Dependencies into Components

Imagine a teacher entering a classroom. They don't bring chalk from home. Instead, the principal (Angular) provides the chalk they need to teach.

In Angular, services work the same way: Angular gives components the tools they need.

Here's how it looks in code:

```
@Component({ ... })
export class ClassroomComponent {
  constructor(private chalk: ChalkService) {
    this.chalk.use();
  }
}
```

The **constructor** is like the teacher's desk. It's where Angular gives the **chalk** (service) to the **teacher** (component).

Where does the chalk come from? The principal prepares it like this: see the code

```
@Injectable({ providedIn: "root" })
export class ChalkService {
  // Chalk-related logic
}

```

@Injectable makes it a service, and **providedIn: 'root'** ensures it's available to the whole app.

Angular handles everything, so components (teachers) can focus on their job!

Singleton Services – One Set of Chalk for All

Imagine a school where **all teachers share the same box of chalk**. Instead of giving each teacher their own box, the school provides one shared box. Teachers take what they need and pass it along.

In Angular, services work the same way when defined as **Singletons**:

- Angular creates only one instance of the service for the entire app.
- All classrooms (components) share the same service, saving resources and ensuring consistency.

Here's the code :

```
1  @Injectable({ providedIn: "root" })
2  export class ChalkService {
3    // Chalk-related logic
4  }
5
```

- **@Injectable({ providedIn: 'root' }):** This makes the chalk service available across the whole app.
- Angular ensures one shared instance of SharedChalkService.

No duplicate supplies = more efficient and consistent !

Customizing Providers – Private Chalkboards for Each Classroom

Imagine a school where every classroom has its own **private chalkboard.**

The chalkboard in one classroom doesn't affect the others. **For example**, the math class can write formulas without worrying about what's on the science class chalkboard.

Angular lets you decide:

1. Share resources globally (like chalk for the whole school).

Keep resources private to one classroom (a specific component).

Here's how you create a private chalkboard ::

```
@Component({
  providers: [PrivateChalkboardService], // Private chalkboard for this class
})
export class MathClassComponent {
  constructor(private board: PrivateChalkboardService) {
    this.board.write("Math formulas");
  }
}
```

In this setup, every **math class** gets its own chalkboard (PrivateChalkboardService). It's **separate** from others, ensuring no overlap with other classes.

Benefits of Dependency Injection

Why Angular's School System is Genius

Angular's **Dependency Injection (DI)** is like a brilliant school system that makes everything work smoothly:

1. **Saves Time:** Teachers (components) don't need to bring their own chalk (services). The principal (Angular) provides it!
2. **Shares Resources:** One chalk box (service) can be shared by many classrooms (components).
3. **Improves Testing:** Want to test the chalk without a real classroom? Easy! Use a pretend chalk box for testing.

```
TestBed.configureTestingModule({
  providers: [{ provide: ChalkService, useClass: FakeChalkService }],
});
```

In this setup, a **FakeChalkService** acts like a pretend chalk box, making testing faster and easier.
Dependency Injection makes Angular apps smarter and more efficient just like a well-organized school.

Angular Tips

Follow me to get more Information and tips like this.

Chandrakanth Avula
@chandrakanth1